Approval Sheet

Title of Dissertation: Semantically-Linked Bayesian Networks: A Framework for Probabilistic Inference Over Multiple Bayesian Networks

Name of Candidate: Rong Pan
Doctor of Philosophy, 2006

Dissertation and Abstract Approved: _____

Doctor Yun Peng
Professor (Chair)
Department of Computer Science
and Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Rong Pan

Permanent address: 20 Casey Ct, Catonsville, Maryland, 21228.

Degree and date to be conferred: Ph.D., 2006.

Date of Birth: June 13, 1979.

Place of Birth: Beijing, People's Republic of China.

Secondary education:

The Fifth High School of Beijing,
Beijing, People's Republic of China, June 1997

Collegiate institutions attended:

Peking University, Beijing, People's Republic of China
September 1997 – June 2001, B.S., June 2001

Major: Computer Science

University of Maryland, Baltimore County, Baltimore, Maryland
September 2001 – August 2006, Ph.D., December 2005

Major: Computer Science

Professional Publications:

Rong Pan, Yun Peng, and Zhongli Ding, *Belief Update in Bayesian Networks Using Uncertain Evidence*. Submitted to the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI06).

Rong Pan, Zhongli Ding, Yang Yu, and Yun Peng, *A Bayesian Network Approach to Ontology Mapping*. In Proceedings of the Fourth International Semantic Web Conference (ISWC2005), Galway, Ireland, November 2005.

Zhongli Ding, Yun Peng, and Rong Pan, *BayesOWL: Uncertainty Modeling in Semantic Web Ontologies.* In Soft Computing in Ontologies and Semantic Web, Springer-Verlag, December 2005.

Li Ding, Rong Pan, Tim Finin, Anupam Joshi, Yun Peng and Pranam Kolari, *Finding and Ranking Knowledge on the Semantic Web.* In Proceedings of the Fourth International Semantic Web Conference (ISWC2005), Galway, Ireland, November 2005.

Li Ding, Rong Pan, Tim Finin, Anupam Joshi, Yun Peng and Pranam Kolari, *Search on the Semantic Web.* IEEE Computer, October 2005, 62-69.

Rong Pan and Yun Peng, *A Framework for Bayesian Network Mapping.* The Twentieth National Conference on Artificial Intelligence (AAAI-05), student abstract, Pittsburgh, PA, July 2005.

Zhongli Ding, Yun Peng, Rong Pan, and Yang Yu, *A Bayesian Methodology Towards Automatic Ontology Mapping.* AAAI-05 Workshop on Contexts and Ontologies: Theory, Practice and Applications, Pittsburgh, PA, July 2005.

Tim Finin, Li Ding, Rong Pan, Anupam Joshi, Pranam Kolari, Akshay Java and Yun Peng, *Swoogle: Searching for knowledge on the Semantic Web.* In proceedings of AAAI 05 (intelligent systems demo) , July, 2005.

Zhongli Ding, Yun Peng, and Rong Pan, *A Bayesian Approach to Uncertainty Modeling in OWL Ontology*. In Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA2004). November 15-18, 2004, Luxembourg-Kirchberg, Luxembourg.

Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Joel Sachs, Vishal Doshi, Pavan Reddivari, and Yun Peng, Swoogle: *A Search and Metadata Engine for the Semantic Web*, Thirteenth ACM Conference on Information and Knowledge Management (CIKM'04), Washington DC, November 2004.

Youyong Zou, Tim Finin, Li Ding, Harry Chen, and Rong Pan, *Using Semantic Web technology in Multi-Agent Systems: a case study*, in the TAGA trading agent environment 5th International Conference On Electronic Commerce: Technologies, Pittsburg, 1-3 October 2003.

Youyong Zou, Tim Finin, Li Ding, Harry Chen, and Rong Pan, *TAGA: Trading Agent Competition in Agentcities*, Workshop on Trading Agent Design and Analysis, held in conjunction with the Eighteenth International Joint Conference on Artificial Intelligence, 11 August, 2003, Acuulco MX.

Professional Positions Held

2003 – 2006:

Research Assistant

Ebiquity Lab, UMBC

2005 (summer)

Summer Intern

Stottler Henke Associates Inc.

2002 (spring):

Intern

Silicon Graphic, Inc., China

Abstract

Title of Dissertation: Semantically-Linked Bayesian Networks: A Framework for Probabilistic Inference Over Multiple Bayesian Networks

Rong Pan, Doctor of Philosophy, 2006

Dissertation Directed by:

Yun Peng
Professor
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County

At the present time, Bayesian networks (BNs), presumably the most popular uncertainty inference framework, are still widely used as standalone systems. When the problem itself is distributed, domain knowledge has to be centralized and unified before a single BN can be created. Alternatively, separate BNs describing related sub-domains or different aspects of the same domain may be created, but it is difficult to combine them for problem solving even if the interdependent relations between variables across these BNs are available. Existing approaches have greatly restricted expressiveness and applicability as they either impose very strong constraints on the distributed domain knowledge or only focus on a specific application. What is missing is a principled framework that can support probabilistic inference over separately developed BNs.

In this thesis, we propose a theoretical framework, named *Semantically-Linked Bayesian Networks* (SLBN), to fill this blank. SLBN is distinguished from existing work in that it defines linkages between semantically similar variables and probabilistic influences are carried by variable linkage from one BN to another by soft evidences and virtual

evidences. To support SLBN's inference, we have developed two algorithms for belief update with soft evidences. Both of these algorithms have clear computational and practical advantages over the methods proposed by others in the past. To justify SLBN's inference process, we propose J-graph to represent the jointed knowledge of the linked BNs and the variable linkages. Finally, SLBN is applied to the problem of concept mapping between semantic web ontologies.

Semantically-Linked Bayesian Network:

A Framework for Probabilistic Inference Over

Multiple Bayesian Networks

by
Rong Pan

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

To my family

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 The Motivations

At the present time, Bayesian networks (BN), the presumably the most popular uncertainty inference framework, are still widely used as standalone systems. When the problem scale is large, a large network slows down the inference process and makes itself difficult to be reviewed or revised. When the problem itself is distributed, domain knowledge has to be centralized and unified before a single BN can be created for the problem. Alternatively, separate BNs describing related sub domains or different aspects of the same domain may be created, but it is difficult to combine them for problem solving — even if the interdependent relations between variables across these BNs are available. This is because, among other things, individual BNs may adopt different variables with identical or similar semantics, and even identical variables may assume different distributions in different BNs.

Much research has been done recently to address this issue, including the most notable Multiply Sectioned Bayesian Network (MSBN) by Xiang [42][43][44] and Agent Encapsulated Bayesian Network (AEBN) by Bloemeke and Valtorta [2][3]. However, these existing approaches still have restrictions in consistency, applicability and expressiveness. For example, in the scenario of MSBN, a global BN is sectioned into subnetworks with shared variables. Each subnet is occupied by an agent and all subnets must be strictly consistent with each other in that the shared variables are identical and all of the parents of a shared variable must appear in one subnet. In essence, the domain

knowledge of MSBN, although distributed in several subnets, is accessed as a single BN in inference. AEBN is proposed for modeling sensor networks, where each agent utilizes a BN to represent its domain knowledge and exchange beliefs with other agents via connections between identical variables. These connected variables are allowed to have different distributions, but only the belief of the evidence side is finally respected at the end of the inference. This is done by treating the exchanged belief as soft evidences when the inference is carried across different BNs. However, AEBN does not address the issue of how to propagate probabilistic influences when multiple connections are available between two BNs, nor does it discuss how the linkages should be created or how to resolve the conflicts caused by multiple connections. Most importantly, none of the above approaches offers solutions for representing probabilistic similarities between variables in probability-theoretic terms, which is required when we want to link similar but not identical variables across BNs. These and other related issues are the focus of our investigation.

Probabilistic reasoning in distributed BN models can be utilized in many applications, such as multi-agent systems, data mining, and sensor networks. Besides, Semantic web also provides a use case. Separately developed Semantic web ontologies on the same or overlapped domain may differ in terminologies and taxonomical organizations, or use polysemantic terms to refer different concepts. Although similar concepts can be easily described by probabilities, existing approaches [13][30][38] fail to completely address uncertainty in concept mappings. In a previous work, Ding, Peng and Pan have developed a Bayesian network based framework, *BayesOWL*, to address representation and reasoning with uncertainty within a single ontology [11][12]. This framework makes it

possible for us to use SLBN to model ontology mappings, and discover new concept mappings by probabilistic inferences.

1.2 Thesis Statement

In this thesis, Semantically-Linked Bayesian Networks (SLBN) is proposed as a theoretical framework to support probabilistic reasoning across multiple BNs. This framework is distinguished from existing work in that the SLBN defines linkages between semantically similar variables and through these linkages belief update can be conducted over separately developed BNs.

The representation of semantic similarities in SLBN is the *variable linkage*, whose role is to carry probabilistic influence from one BN to another. The probabilistic influences from other BNs, due to their nature of uncertainty, are treated as soft evidences and virtual evidences. Since inference with soft evidences does not obey Bayes' rule, two algorithms are developed in this thesis for belief update with uncertain evidences. These algorithms, which incorporate Jeffrey's rule for conditioning and IPFP (iterative proportional fitting procedure) for modification of joint distributions, are further expanded to respect the semantics of variable linkages and to support the inference of SLBN.

A graphical probabilistic model, J-graph, is proposed to describe the jointed knowledge of SLBN. Among other things, J-graph can be used to study the consistency of SLBN and help to justify the inference of SLBN.

SLBN is designed to support various applications. In the last part of this thesis, SLBN, together with BayesOWL [11][12], is applied to discover concept mappings in semantic

web ontologies. Experiments show that this approach discovered new complex concept mappings that existing approaches are unable to find.

1.3 Dissertation Outline

This dissertation is outlined as follows. In Chapter 2, we introduce the background of this work, including Bayesian networks and the Semantic Web technology, and briefly review the previous researches related to this work.

Chapter 3 discusses the relations between existing belief update methods for uncertain evidence and proposes two soft evidential update methods, which integrates traditional BN inference methods, IPFP and virtual evidences. These soft evidential update methods will be used in the SLBN inference methods.

In Chapter 4, we define semantic similarities between concepts and use variable linkages to represent them in BN. We also discuss the expressiveness of variable linkages in this chapter.

Chapter 5 presents SLBN's inference method. First, we give an informal description about how the probabilistic influences should be propagated and then detailed implementations are provided by pseudo codes.

Chapter 6 discusses about the global knowledge of SLBN. Since an SLBN should have a global knowledge, three assumptions are proposed to ensure the linked BNs and the linkages in an SLBN are consistent to the global knowledge. J-graph is then proposed to represent the jointed knowledge of the linked BNs and the variable linkages. Finally, J-graph is used to justify the inference method of SLBN.

In Chapter 7, as an application example, we use SLBN to discover concept mappings between different semantic web ontologies. Experiments are provided in this chapter.

Chapter 8 concludes this dissertation and lists some interesting issues for future research.

We adopt the following notations in this thesis. A BN is denoted as N. X , Y, and Z are for sets of variables in a BN, and x or x_i are for a configurations of the states of X. Capital letters A, B, C are for single variables. Capital letters P, Q, and R, are for probability distributions; the subscription of P, Q, and R denotes the origin of the probability distribution. V is for a set of nodes in graphs.

2 Background and Related Work

The Bayesian network was first introduced to the AI community by Pearl [33][35] as a graphic model to represent the uncertainty of domain knowledge and to reason with such uncertain knowledge. The exact inference method of BN has been proved to be NP hard [7], and exact inference methods, such as Junction-Tree [26], are thus of exponential time complexity. Approximate inference methods, such as loopy propagation [28] and stochastic simulation [33], are invented for reasoning with large scale networks.

The distributed model of BN is investigated for large scale and distributed problems. Multiply sectioned Bayesian network (MSBN) by Xiang [42][43][44] and Agent Encapsulated Bayesian Network (AEBN) by Bloemeke and Valtorta [2][3] are two prominent examples. MSBN aims to section one large BN into smaller subnets and deploy them to agents. The consistency of cross subnet linkage is guaranteed by the sectioning method. AEBN aims to propagate the beliefs between separate agents and the consistency of variables' distributions is ensured by the system's semantics.

Another background knowledge related to our proposed research is Semantic Web [36], which comes with the rapid expansion of computer network and the World Wide Web and the need for machines/programs to understand web page content. Semantic web describes information resources using URI and XML namespace mechanism, which is formalized as the Resource Description Framework (RDF) [47]. The vocabulary of semantic web resources is defined as a set of ontologies in the format of RDF schema. Since multiple ontologies are often involved, ontology mapping and translation become

necessary for information integration in semantic web. Most existing works in this direction are logic based [14][18], which ignore the uncertainty in the mapping. A few exceptions include GLUE [13] from the University of Washington and OntoMapper [38] from UMBC, which have addressed uncertainty to some extent.

As current ontology description languages do not capture all the relations between concepts, uncertainty extension of ontology has been an interesting research issue. Ding [11][12] proposed a framework called BayesOWL which extends OWL to allow expression of probability information and then converts ontologies to BNs by a set of translation rules and procedures. Holi's work similarly aims to approximate the concept overlaps of a semantic web taxonomy using Bayesian network [17].

2.1 Bayesian Network

Bayesian network (BN) is an increasingly popular knowledge representation framework for uncertainty. It provides a systematic way to represent interdependency relationships among propositions, each of which is considered as a random variable with a finite set of states as its outcomes. Intense research on BN has been conducted in recent years, as reflected by a rich body of publications in the literature [45]. In what follows, we briefly review the basics of BN and techniques relevant to our research agenda.

Let A and B be two propositions, normally we use $P(A)$ to denote unconditional or *prior probability*, $P(A|B)$ the conditional or *posterior probability*. $P(V)$ denotes the Joint Probability Distribution (JPD) of the entire domain. If X is a subset of V, $P(X)$ is referred to as a *marginal distribution*, and from JPD we can calculate $P(X)$ using the following formula

$$P(X) = \sum_{V \setminus X} P(X, V \setminus X) . \quad \textbf{(2.1)}$$

The theoretical basis of Bayesian network is Bayes's rule [1], which underlies all probabilistic inferences in BN:

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}, \quad \textbf{(2.2)}$$

where A and B are propositions.

Definition 2.1 (Bayesian network [19]): A Bayesian network is a graph in which the following holds:

1. A set of variables and a set of directed edges between variables.
2. A finite set of mutually exclusive states for each variable.
3. A directed, acyclic graph (DAG).
4. A conditional probability table (CPT) for each variable that quantifies the effects of its parents.

As can be seen shortly, each Bayesian network corresponds to a particular probability distribution over all variables of a BN. One the other hand, however, one probability distribution can be represented by many BNs with the same variables but different topological structures.

In this thesis we use $d(A)$ **to denote the number of the states for variable** A**, and the total number of the state configurations for a set of variables** X **is calculated as** $d(X) = \prod_i d(X_i)$ **. An edge from variable** A **to** B **is denoted by** $A \rightarrow B$**, and a directed**

path from variable *A* **to** *B* **is denoted by** *path*(*A*, *B*)**, a path from** *A* **to** *C* **via** *B* **is denoted by** *path*(*A*, *B*, *C*)**.**

Let $x = \{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\}$ be a set of propositions in a Bayesian network, where $\{X_1, X_2, \ldots, X_n\}$ is the set of the variables in the BN. The chain rule for Bayesian network is described as below:

$$P(x) = \prod_i P(x_i \mid parent(X_i)), \quad \textbf{(2.3)}$$

The chain rule is based on a conditional independence assumption associated with BN, which can be described by the notion of *d-separation* in terms of the network's graphic structure. Two distinct variables *A* and *B* are said to be d-separated if, for all paths between *A* and *B*, there is an intermediate variable *V* such that either

1. the connection through *V* is serial or diverging and V is instantiated, or
2. the connection is converging, and neither *V* nor any of *V*'s descendants have received evidence.

If *A* and *B* are not d-separated, we call them *d-connected*. In a BN, if *A* and *B* are d-separated, they are in independent of each other. In other words, if *A* and *B* are d-separated, the changes in the certainty of *A* have no impact on the certainty of *B* [19].

The *Markov blanket* of a variable refers to the variable's parents, children and children's other parents. If all variables in the Markov blanket are instantiated, the Markov blanket of variable *A* d-separate *A* from the rest of the BN, and hence *A* is independent of any other variables. Let p_A denote the parents of *A* and l_A denote the

children and children's other parents of A, $M_A = p_A + l_A$. Figure 2.1 depicts an example of Markov blanket.

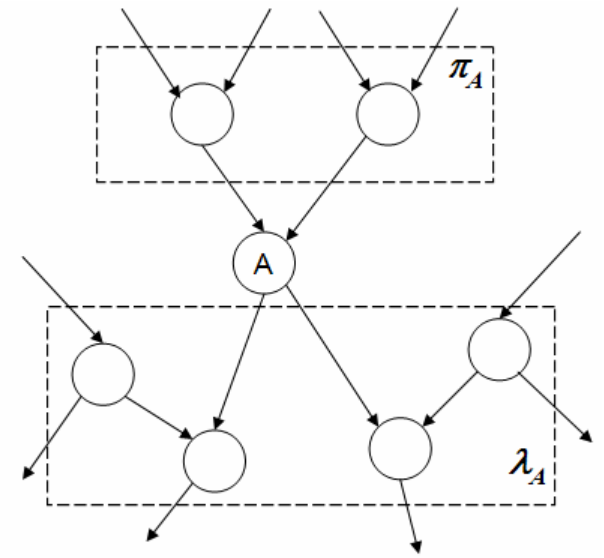


Figure 2.1 An example of Markov Blanket

It has been proved that exact inference in Bayesian network is NP-hard [7]. Nonetheless, a number of algorithms have been developed to exploit the network structure for efficient computation for probabilistic inferences in BN. Belief propagation [35] and Junction-Tree [26] are two of the most popular exact BN inference methods. Belief propagation is based on local message passing and hence can work only for polytree to avoid rumors (double counting probability influence in belief propagation). Junction-Tree method, on the other hand, works for networks of any kinds of topology. Several inexact algorithms are developed for large scale problems with approximate solutions (e.g., stochastic sampling [34] and loopy propagation [28]).

Junction-Tree method, which will serve as a basis for developing some of our problem solving algorithms in this research, is briefly reviewed here. This method

transforms a Bayesian network into a tree structured graph of large nodes. It first moralizes the network by adding edges between nodes that share children and removing the directions of all links. Then, the moral graph is triangulated so that all cycles longer than 3 have a chord. In the triangulated graph, each complete subgraph is recognized as a *clique*. Cliques are connected, forming a *Junction-Tree*. The probabilities of cliques are called potentials, on which operations are defined as a set of algebra. Inference is done by passing and unifying potentials between adjacent cliques in the Junction-Tree. Four procedures are used for local message passing: **Absorption**, **DistributeEvidence**, **CollectEvidence**, and **EnterEvidence**. Absorption specifies the method to unify the potentials of adjacent cliques. CollectEvidence calls adjacent cliques and absorbs from them after the call returns. DistributeEvidence absorbs from the clique that calls it and then call other adjacent cliques. EnterEvidence changes the potentials of a clique to represent an observation. Compared with other exact inference methods, Junction-Tree inference method is efficient in that its computation cost is exponential to the size of the largest clique, not the size of the entire network.

For Junction-tree and other belief update methods mentioned above, evidences are instantiations of variables. However, not all evidences are of certainty. Belief update in Bayesian network with uncertain evidences will be discussed in Chapter 3 where we introduce our own new method which supports inference algorithms of SLBN.

Some methods for belief update with uncertain evidences, including our own, utilize the IPFP procedure, which is briefly reviewed next.

2.2 Iterative Proportional Fitting Procedure

Iterative proportional fitting procedure (IPFP) is a mathematical procedure that modifies a joint distribution to satisfy a set of probability constraints, which are distributions of subset of variables. IPFP was first published in [22], and shortly after was proposed as a procedure to estimate cell frequencies in contingency tables under some marginal constraints [10]. Csiszar [6] provided a convergence proof for IPFP based on *I-divergence* geometry. Vomlel rewrote a discrete version of this proof in his PhD thesis [41]. IPFP was extended in [4] and [9] as conditional iterative proportional fitting procedure (CIPFP) to also take conditional distributions as constraints, and the convergence of CIPFP was established for the finite discrete case.

I-divergence (also known as *Kullback-Leibler distance* or *cross-entropy*) is a measure of the distance between two joint distributions P and Q over X:

$$I(P \parallel Q) = \begin{cases} \sum_{P(x)>0} P(x) \log \frac{P(x)}{Q(x)} & \text{if } P << Q \\ +\infty & \text{otherwise} \end{cases} \qquad \textbf{(2.4)}$$

where $P << Q$ means Q dominates P (i.e., $\{x \mid P(x) > 0\} \subseteq \{x' \mid Q(x') > 0\}$). $I(P \parallel Q) \geq 0$ for all P and Q, the equality holds only if $P = Q$.

A *probability constraint* $R(Y \subseteq X)$ to distribution $P(X)$ is a distribution on Y. We say $Q(X)$ is an I_1*-projection* of $P(X)$ on a set of constraints $\boldsymbol{R}$ if $I(P \parallel Q)$ is smallest among all distributions that satisfy $\boldsymbol{R}$.

For a given distribution $Q_0(X)$ and a set of *consistent*[1] constraints $\boldsymbol{R}$ = {$R(Y_1)$, $R(Y_2)$, …, $R(Y_m)$}, IPFP converges to $Q^*(X)$ which is an I_1-projection of $Q_0(X)$ on $\boldsymbol{R}$. $Q^*(X)$, which is unique for the given $Q_0(X)$ and $\boldsymbol{R}$, can be computed by iteratively modifying the distributions according to the following formula, each time using one constraint in $\boldsymbol{R}$:

$$Q_k(X) = \begin{cases} 0 & \text{if } Q_{k-1}(Y) = 0 \\ Q_{k-1}(X) \cdot \dfrac{R(Y_i)}{Q_{k-1}(Y_i)} & \text{otherwise} \end{cases}$$

where m is the number of constraints in $\boldsymbol{R}$, and $i = ((k-1) \bmod m) + 1$ determines the constraint used at step k. For clarity, in the rest of this paper, we write the above formula as

$$Q_k(X) = Q_{k-1}(X) \cdot \frac{R(Y_i)}{Q_{k-1}(Y_i)} \qquad \textbf{(2.5)}$$

The convergence of IPFP is proved by Csiszar [6] and Volmel [41] on discrete distributions. For the continuous case, it still remains as an open question. An important theorem in Volmel's proof ([41] Theorem 2.5) is re-stated below:

Theorem 2.1: Let Q_0 be the initial distribution on variables set X, C_1, .., C_n be a set of probability constraints on X, S_1, …, S_n be sets of joint probability that satisfy the constraints C_1, .., C_n respectively, and S = n S_i ? Ø. Define Q_i recursively by letting Q_i be the I_1-projection of $Q_{i\text{-}1}$ on C_j, where j = (i - 1) mod n + 1. Then the sequence of probability distributions Q_n converges to the I_1-projection of Q_0 on {C_1, …, C_n}.

[1] A set of constraints $\boldsymbol{R}$ is said to be consistent if there exists a distribution $Q(X)$ that satisfies all R_i in $\boldsymbol{R}$. Obviously, two constraints are inconsistent if they give different distributions to the same variable. More discuss of this matter is given in Section 3.7.

We will use this theorem to prove the convergence of our soft evidential update method later in Chapter 2.

2.3 Distributed Bayesian Network Models

Inference across multiple intelligence systems involving uncertainty has been a concern of the AI community for over ten years. Distributed Bayesian networks models are mostly considered in various multi-agent scenarios. An agent is commonly defined as an individual software and/or hardware entity that responds automatically to the environment. In a multi-agent system, each agent has only partial knowledge or a specific perspective of the domain, pursues a local set of goals, and exchanges its observations or beliefs with other agents in the system. If the domain (or some of its aspects) is modeled by a Bayesian network, the network shall be distributed in some fashion among these agents. The major issues of this area have been addressed in the applications of data mining, multi-agent system and sensor networks are as follows:

1. How to decompose the network among the agents?

2. How to exchange local beliefs and observations via agent communications?

3. How to maintain the global consistency in the system?

Several approaches have been proposed for distributed BN in multi-agent scenarios, including AEBN (Agent Encapsulated BNs), MSBN (Multiply Sectioned BN), and object oriented BN [21]. Each of these approaches makes its own assumptions either to circumvent or solve the aforementioned issues. For example, in AEBN, the Bayesian networks are originally encapsulated in each agent while some variables are shared

between agents. By assuming additional independence relations beyond the definition of Bayesian networks, agents use soft evidence to propagate beliefs of shared variables. Global consistency of AEBN is certified by detecting and eliminating the *rumor problem*. In MSBN, the primary objective is to decompose a given global network into sub networks. The global Bayesian network must conform to a hyper-tree structure to be *soundly sectioned*. The message passing procedures between sections are processed locally. Global consistence is certified by the sectionable graph structure and message passing strategy. These two representative approaches are reviewed in details in the next subsections.

2.3.1 Multiply Sectioned Bayesian Network (MSBN)

Multiply Sectioned Bayesian Network by Xiang [42][43][44] is a distributed multi-agent probabilistic reasoning system that divides a large Bayesian network into multiple sections and distributes them to agents. Xiang starts with five basic assumptions on ideal knowledge representation formalisms for such kind of systems:

1. *Each agent's belief is represented by probability.*
2. *An agent can in general influence the belief of each other agent through direct or indirect communication but can only communicate directly to another agent with shared variables.*
3. *A simpler agent organization is preferred.*
4. *A DAG is used to structure each agent's knowledge.*
5. *Within each agent's sub domain, a JPD is consistent with the agent's belief. For shared variables, a JPD supplements an agent's knowledge with others'.*

Following these assumptions, MSBN conforms to a hyper-tree multiple sectioned directed acyclic graph (MSDAG) structure. The definition of hyper-tree and hyper-tree MSDAG are listed below[44]:

Definition 2.2 (MSBN): let $G = (V, E)$ be a connected graph sectioned into sub graphs $\{G_i = (V_i, E_i)\}$. G_is are organized as a connected tree $?$, where each node is labeled as G_i and each link between G_k and G_m is labeled by the interface V_kn V_m such that for each i an j, V_in V_j is contained in each sub graph on the path between G_i and G_j in $?$. Then $?$ is a **hypertree** over G. Links between the sub graphs of $?$ are called hyperlinks. A hyper-tree $?$ is a **hypertree MSDAG** if for each shared node, there is one and only sub graph G_i that contains all its parents.

A hyper-tree MSDAG is transformed into a tree or forest of a set of Junction-Trees. The messages passed between sectioned BNs contain joint probabilities on hyperlink labels. Xiang developed a protocol analogous to Junction-Tree message passing for sectioned Bayesian networks to coordinate message passing and efficiently synthesizing correct beliefs from the messages.

Compared with AEBN, MSBN has three advantages. First, it does not introduce any additional relations between the variables besides the chain rule of the Bayesian network. Second, the interdependence relation of variables still follows d-separation in the composition of sections. While in AEBN, decedents could not affect the variables' beliefs in ascendants. Third, MSBN does not need to concern itself with the rumor problem since its global structure is a DAG.

The limitations of MSBN are also obvious. To make a composition of MSBN sound, MSBN introduces some restrictions into sectioning, some of them are very restrictive.

First, the local network's belief on shared variables may not be consistent with their global values. To solve this, MSBN requires a great communication cost between the sub graphs. Second, distribution of domain knowledge is forced on shared variables: a hyper-tree MSDAG requires that all parents of a shared node appear in one and only one sub graph. This limits the expressiveness of MSBN in describing distributed knowledge.

Xiang does not emphasis the autonomous attribute in its agent settings. Valtorta and Kim [40] note that it seems that "MSBNs were introduced as a method for parallel distributed inference within a single Bayesian network" rather than for modeling uncertainty for autonomous software agents.

2.3.2 Agent Encapsulated Bayesian Network (AEBN)

Agent Encapsulated Bayesian Network, proposed by Bloemeke and Valtorta [2][3][20], is a system in which each agent employs a Bayesian network as its internal knowledge representation of the domain, and exchanges beliefs with each other along shared variables. The variables of an encapsulated Bayesian network are divided into three groups: 1) those that are only used within the agent (*local set*), 2) those that other agents have better knowledge of (*input set*), and 3) those that currently has better knowledge than other agents (*output set*). An agent may want to get a better knowledge of the nodes in the input set from the *publisher* agent, and if incoming information is void, it assigns estimated values to them. An agent shares the distributions of nodes in the output set to *subscriber* agents via soft evidences. The details of the soft evidential update method that AEBN utilizes is given in Section 3.4.

To update an agent's distribution $P(V)$ with new soft evidences $Q(se_1)$, $Q(se_2)$, … , $Q(se_n)$ for a set of variables $X = \{ X_1, X_2, \ldots, X_n \}$, the following formula is used:

$$Q(V) = P(V \setminus X \mid se_1, se_2, ..., se_n) \cdot Q(se_1, se_2, ..., se_n)$$

$$= \frac{P(V \setminus X)}{P(se_1, se_2, ..., se_n)} \cdot Q(se_1, se_2, ..., se_n),$$

where $Q(se_1, se_2, ..., se_n)$ is the I_1-projection of $P(X)$ on $Q(se_1)$, … , $Q(se_n)$.

From above, we may see AEBN extends BN by imposing additional interdependence relations beyond the semantics of BN. Specifically, when we look at the global network, some parent nodes are not influenced by its descendent nodes' states if they are in the input set since their values are fixed by the soft evidences. In AEBN, relations between agents are asymmetric.

To maintain the global consistence, agents must find the redundant influences introduced by the circle connections and eliminate them. This issue is called the *rumor problem*. Figure 2.2 shows an example of AEBN, where four agents communicate using the shared variables S_1, R_1 and R_2. Agents $Observer_1$ and $Observer_2$ receive the distribution of variable S_1 from the *Sensor* agent, and send out their observations to the *Supervisor* agent, the influence of the *Sensor* agent is doubled in the *Supervisor* agent. The final result L is computed using the following formula:

$$P(L) = \sum_{R_1, R_2} P(L \mid R_1, R) P(R_1) P(R_2),$$

and by considering the observers' local network, we get

$$P(L) = \sum_{R_1, R_2} P(L \mid R_1, R)[P(R_1 \mid S_1) P(S_1)][P(R_2 \mid S_1) P(S_1)]$$

where $P(S_1)$ is double counted.

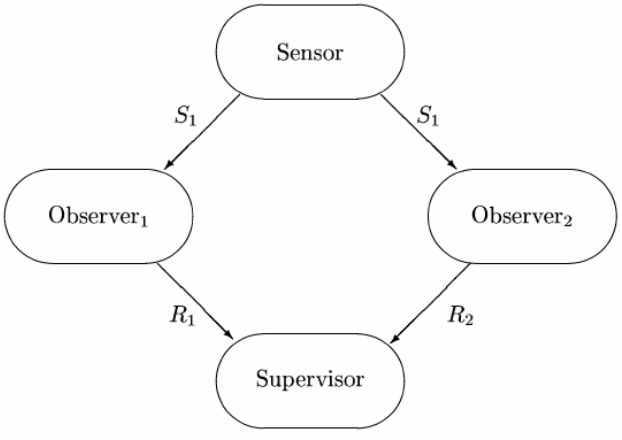


Figure 2.2 ROSE communication graph

Bloemeke [3] proposed two algorithms to remove this kind of redundant influences. The first approach utilizes flow network to discover where this kind of redundant influences appear and then eliminate them by compensating the repeatedly counted propositions from preceding agents. The second approach is a centralized algorithm that involves automatic construction of an auxiliary three-layered local Bayesian network. By carefully manipulating the CPTs of this local Bayesian network and propagating beliefs, distributions of the first layer nodes are fed back to the original distributed networks.

AEBN has limitations in expressing more complicated relations between variables. Also it does not address the issues of consistency when multiple connections between two BNs are present. It only discusses the rumors between networks, not the rumors between the connections. AEBN is designed for a specific application domain and so it is not justified in a principled manner, nor is its applicability well addressed.

2.4 Semantic Web

The current World-Wide Web contains vast volumes of data, of which the majority are web pages organized for human consumption only. Machines and programs understand and process the information provided by such documents. Semantic Web is an attempt to solve this problem by associating meanings/semantics with the data through carefully defined ontologies of the concepts in a way machines can understand.

The Resource Description Framework (RDF) [47][48][49], a collaborative effort by a number of metadata communities, is a standard general assertional model to represent the resources on the web. It is a framework that supports resource description and metadata for a variety of applications. RDF uses XML as its syntax and identifies resources by using URI and XML namespace mechanisms. The basic building blocks of RDF, RDF triples, are represented as “subject”, “predicate” and “object”. The “predicate” is also known as the property of a triple. In general, a triple can be read as “the <subject> has <predicate> <object>” [48]. RDF is an assertional logic, in which each triple expresses a simple proposition [49]. One triple does not change the meaning of other triples.

RDF Schema [50] is used to control the vocabulary used to define the resources in RDF. Classes and properties are created as descriptions of concepts and relations. Property can describe a relationship between classes and properties, restrictions to one property’s domain and range, and annotations to any RDF terms. However, RDF does not specify any inference method. Built on top of RDF and RDF Schema, Web Ontology Language (OWL) [52] provides a richer set of vocabulary to describe the resources and their relations, and its semantics supports description logic inferences.

Ontology is a specification of conceptualization. More formally, ontology is a set of vocabulary describing the conceptualization of a particular domain [46]. It provides a common understanding about the domain knowledge. For a semantic web application, ontology is used to capture the concepts and the relationships between concepts. Since knowledge about its domain is encoded by this ontology, machines or agents then can reason about the domain and adjust their future behaviors accordingly.

OWL [52] is a new web ontology language recommended by W3C. An OWL ontology has zero or more headers, followed by zero or more classes, properties and instances. OWL assigns specific meaning to certain RDF triples using OWL vocabulary. OWL classifies the instances into two sorts: the datatype instances consist of the values that belong to XML Schema datatypes, the object instances consist of individual objects that are instances of classes described within OWL or RDF. Correspondingly, there are two kinds of properties in OWL: ObjectProperty and DatatypeProperty. OWL includes three versions of increasingly complex languages: OWL-Lite, OWL-DL and OWL-Full.

2.4.1 Probabilistic Extension of OWL

The OWL specification does not include any principled means to represent and inference with uncertainty information. However, uncertain information can by found in almost every aspect of ontology engineering. For example, in *domain modeling*, besides knowing that "*A* is a subclass of *B*", one may also know and wish to express that "*A* is a small subclass of *B*"; or, in the case that A and B are not logically related, one may still wish to express that "*A* and *B* are largely overlapped with each other". In ontology reasoning, one may want to know not only if *A* is a subsumer of *B*, but also how close *A* is to *B*; or, one may want to know the degree of similarity even if *A* and *B* are not

subsumed by each other. Moreover, a description (of a class or object) one wishes to input to an ontology reasoner may be noisy and uncertain, which often leads to generalized conclusions. Uncertainty becomes more prevalent in concept mapping between two ontologies where it is often the case that a concept defined in one ontology can only find partial matches to one or more concepts in another ontology.

BayesOWL ([11][12]) is a framework which augments and supplements OWL for representing and reasoning with uncertainty based on Bayesian networks. This framework provides a set of rules and procedures for direct translation of an OWL ontology into a BN structure (a directed acyclic graph or DAG) and a method based on IPFP that utilizes available probability constraints about classes and interclass relations in constructing the conditional probability tables (CPTs) of the BN. The translated BN, which preserves the semantics of the original ontology and is consistent with the probabilistic constraints, can support ontology reasoning, both within and across ontologies, as Bayesian inferences.

Structural translation The general principle underlying the structural translation rules is that all classes (specified as "subjects" and "objects" in RDF triples of the OWL file) are translated into nodes in BN, and an arc is drawn between two nodes in BN if the corresponding two classes are related by a "predicate" in the OWL file, with the direction from the superclass to the subclass.

The model-theoretic semantics of OWL treats the domain as a non-empty collection of individuals. If class A represents a concept, the node it is translated to is treated as a binary random variable of two states a and $\bar{a}$, and we interpret $P(A=a)$ as the prior probability or one's belief that an arbitrary individual belongs to class A, and $P(a|b)$ as

the conditional probability that an individual of class B also belongs to class A. Similarly, for $P(\overline{a})$, $P(\overline{a}\,|\,b)$, $P(a\,|\,\overline{b})$, and $P(\overline{a}\,|\,\overline{b})$, we interpret the negation as "not belonging to".

Control nodes are created during the translation to facilitate modeling relations among class nodes that are specified by OWL *logical* operators, and there is a converging connection from each of the concept nodes involved in this logical relation to its specific control node. There are five types of control nodes in total corresponding to the five types of logical relations: "and" (owl:intersectionOf), "or" (owl:unionOf), "not" (owl:complementOf), "disjoint" (owl:disjointWith), and "same as" (owl:equivalentClass).

Constructing CPTs The nodes in the DAG obtained from the structural translation step can be divided into two disjoint groups: *XR*, regular nodes representing concepts in ontology, and *XC*, control nodes for bridging logical relations. The CPT for a control node in *XC* can be determined by the logical relation it represents so that when its state is "True", the corresponding logical relation holds among its parent nodes. When all the control nodes' states are set to "True" (denote this situation as *CT*), all the logical relations defined in the original ontology are held in the translated BN. The remaining issue is then to construct the CPTs for node in *XR* so that $P(XR|CT)$, the joint distribution of all regular nodes in the subspace of *CT*, is consistent with all the given probabilistic constraints about classes and relations between classes. These constraints include, most likely, priors for classes $P(C)$, conditionals $P(C|D)$ for relations between classes *C* and *D*. Several suggestions have been made to encode probability constraints in semantic web languages (e.g., [15] with RDF). These constraints can be obtained from the ontology

designers or learned from data (an approach that learns these constraints from web is described in Section 7.1).

In principle, IPFP can be applied to construct CPTs to satisfy all the given probabilistic constraints. Two difficulties exist. First, direct application of IPFP may destroy the existing interdependencies between variables (i.e., the given DAG becomes invalid). Secondly, IPFP is computationally very expensive since every entry in the joint distribution of the BN must be updated at each iteration. To overcome these difficulties, Peng and Ding [37] developed an algorithm named D-IPFP that decomposes IPFP so that each iteration only updates a small portion of the BN that is directly involved with the chosen constraint, and the update is done only to CPTs while keeping the DAG of the network intact [37]. In particular, when each of the given constraints involves only one variable C_i and a set of zero or more of its parents S_i, (2.5) of IPFP becomes [12]:

$$\begin{cases} Q_k(C_i \mid p_i) = Q_{k-1}(C_i \mid p_i) \cdot \dfrac{Q(C_i \mid S_i)}{Q_{k-1}(C_i \mid S_i)} \\ Q_k(C_j \mid p_j) = Q_{k-1}(C_j \mid p_j) \end{cases} \forall j \neq i .$$

The *BayesOWL* framework can support common ontology reasoning tasks as probabilistic inferences in the translated BN. For example, given a concept description *e*, it can answer queries about concept satisfiability (whether *P(e|CT)* = 0), about concept overlapping (how close *e* is to a concept *C* as *P(e|C,CT)*), and about concept subsumption (find the concept which is most similar to *e*) by defining some similarity measures such as Jaccard coefficient [39].

PR-OWL [8] is another probabilistic extension of OWL based on Multi-Entity Bayesian Network (MEBN) [25]. Different from BayesOWL, which utilizes BN's

propositional logic expressiveness, PR-OWL can express probabilistic models on first-order logics level. To represent probabilistic information, PR-OWL provides an upper ontology of uncertainty notations in OWL syntax. PR-OWL is currently in a preliminary stage and examples are yet to present.

2.4.2 Information Integration in Semantic Web

Separately developed Semantic Web ontologies on the same or overlapped domain may differ in terminologies and taxonomical organizations, or use polysemantic terms to refer to different concepts. Ontology mappings are hence required to mark up the related concepts in different ontologies. And, if possible, we want to translate the instances of classes from one ontology to another using the mappings.

Current approaches for concept mapping between ontologies mostly use heuristics from linguistic analysis of concept names and concept's literal descriptions, heuristic from language specific taxonomy structures, or machine-learning algorithms to discover relativity between concepts from separate ontologies. These approaches use either numerical measurements of concept similarities or logic expressions to describe discovered mappings. When using numerical similarities for concept mapping discovery, the similarities can be obtained from machine learning algorithms or supervised learning algorithms. Text based learning algorithms or heuristics are widely used to calculate the similarities using the text descriptions of concepts or sets of exemplars attached to the concepts. Ontologies' the local structure info is also utilized by some to help improve the accuracy. Finally similarities are filtered by hard thresholds to judge if two concepts are identical. Similar but not identical concepts are not respected by these approaches. Most processes utilizing numerical similarities are automatic or semi-automatic. To logically

encode ontology mappings, logic axioms are used to model the relations between the concepts and these axioms are created manually by domain experts. Logic can encode more expressive and complex mappings than numerical similarities but is hard to be automatically created.

GLUE [13] is a successful system by the University of Washington in exhibiting a standard model of similarity based ontology mapping system. Given one concept in ontology A, GLUE aims to calculate the similarity of this concept and all other concepts in ontology B, and find the most similar one. GLUE is capable of working with different kinds of similarity measurements. It applies multiple machine learning algorithms and uses a meta learner to combine the learning results. Learning results are represented as joint probabilities, which are further translated into a similarity matrix by applying similarity measurements. Finally, a Relaxation labeler module takes the similarity matrix along with domain constraints and other heuristics knowledge to search for the best mapping configuration. GLUE uses text, practically a set of instance documents of concept and the concept's name, as raw data for similarity calculation. Naïve Bayes learning method is used to classify instance documents into four sets: $P(A,B)$, $P(A,\overline{B})$, $P(\overline{A},B)$, and $P(\overline{A},\overline{B})$ and a set of joint probabilities is calculated based on these sets.

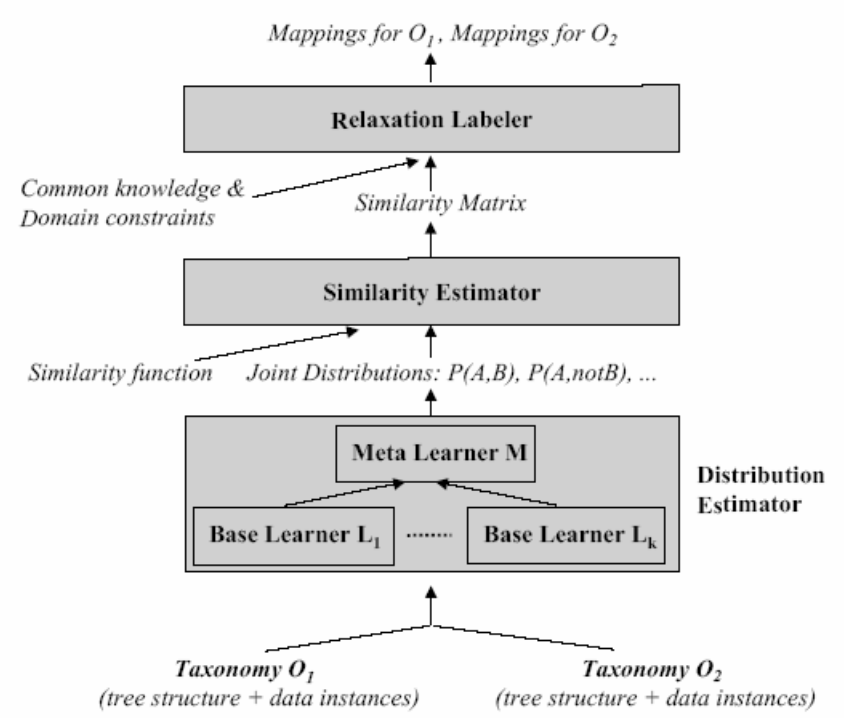


Figure 2.3 Architecture of GLUE

OntoMapper [38] is another similarity approach by UMBC, which improves the text-based classification result by conducting probability reasoning using local structure. First, raw similarity scores between concept pairs is obtained by using the Rainbow classifier [39] system on a set of instance documents. Then, two algorithms are provided to refine the results. The first one is a simple heuristic method which realizes subsumption based on the majority rule: for any non-leaf node, if the ratio of its children that map to a particular node is greater than or equal to a user-specified threshold, then these children's mappings along with the values associated are propagated up to the parent node. The second one is a probabilistic approach based on Bayesian reasoning, in which some heuristics are used to respect the structures of the taxonomies.

OntoMerge from Yale [14][51] is an online ontology translation system which translates an ontology A into a new DAML+OIL ontology B that captures the same semantic information. It is built on top of PDDAML (PDDL-DAML Translator) [54] (based on Jena [53]) and OntoEngine (an inference engine) (based on JTP [55]). Users need to first specify the relations between mapped variables using logical expressed *bridging axioms*. Axioms use vocabulary from items from both source and target ontologies and use namespaces to avoid duplicate names. At first, OntoMerge calls PDDAML to translate ontology B into a Web-PDDL file, and then feeds this file to OntoEngine. OntoEngine retrieves a merged ontology C from its library which covers A and B. OntoEngine then tries to load ontology C in by first using PDDAML to translate it to Web-PDDL file. At last, OntoEngine loads A in, translates it, and calls PDDAML again to translate the Web-PDDL results back into DAML+OIL. The merged ontology C is obtained by taking the union of the terms and axioms defining them. Bridging axioms that relate the terms in A to the terms in B through the terms in C should also be added. Since all the axioms used are specified by human experts, this approach is semi-automatic.

PROMPT from Stanford [30] is a user-interactive methodology of ontology mapping and alignment. PROMPT provides mapping suggestions based on linguistic analysis of the given ontologies, and asks users to select an operation. Then PROMPT applies user's operation, identifies conflicts between current operation and previous operations and update its suggestions. Conflicts are identified by a set of heuristic from domain experts.

2.5 Summary

In this chapter, we presented the background of this thesis research and related techniques in the areas of Bayesian networks and Semantic Web.

As we can use BNs to represent semantic web ontologies, distributed BN models can further help to conduct probabilistic reasoning across multiple semantic web ontologies, and hence help with the ontology mapping problem. However, existing approaches of distributed Bayesian network models, such as MSNB and AEBN, are not satisfactory for such purposes. MSBN is designed for decomposing global domain knowledge into pieces so that they can be deployed into agents. AEBN is proposed in the scenario of sensor networks and is not well justified for general purposes. In both MSBN and AEBN, similar but not identical variables cannot be modeled to link separate BNs.

Moreover, semantic web ontologies are mostly separately developed and deployed. Two ontologies may be defined on the same domain but represent different conceptualization of that domain, or capture different aspects of the domain, and use different terminologies. Also ontologies of different but relative domains may share common or similar concepts, and finding and modeling the shared concepts is crucial to collaborative inference using different ontologies. The existing ontology mapping approaches are aimed at finding simple identical mappings, while the OWL syntax actually defines complex relations between concept classes, including equivalent, union, disjoint, and intersection.

3 Belief Update in Bayesian Network Using Uncertain Evidence

As will be seen later in Chapters 4 and 5, inference across different BNs in our new framework shall be carried out by exchanging probabilistic influences between them via variables of the same or similar semantics. In BN, these probabilistic influences can be conveniently represented as external findings, named as *uncertain evidential findings*. Here the term "uncertainty" refers to the fact that these influences are modeled as probability distributions over these variables, not instantiations to their states. In this chapter, we consider the problem of updating beliefs in Bayesian Networks (BN) by uncertain evidential findings. The algorithms developed in this chapter will serve as a basis for probabilistic inference in our SLBN.

It is easy to tell hard evidence from uncertain evidence. However, the difference between soft and virtual evidence, and even their exact semantics, are still subject for debate. We take the following in this paper: for a soft evidence, the observation (e.g., $Q(A)$) is to be believed with certainty although this observation itself is about the uncertainty of the states A is in. In contrast, for a virtual evidence, we are uncertain about the observations themselves. Consequently, soft evidence (e.g., $Q(A)$) should be preserved in the updated belief (or put in other words, the original joint distribution of the BN should be modified to be consistent with $Q(A)$). On the other hand, reasoning with virtual evidences requires preservation of only the given likelihood ratios, not the probabilities of evidential variables. This distinction between soft and virtual evidences is

crucial in understanding our proposed new methods, and it will be further discussed later in Section 5.

There are three main methods for revising the beliefs of a BN with uncertain evidence: *virtual evidence method* [35], *Jeffrey's Rule* [18], and *iterative proportional fitting procedure* (IPFP) [41]. The virtual evidence method, as its name indicates, requires the evidential findings to be in the form of likelihood ratios, while Jeffrey's rule works with evidential findings in the form of probability distributions and thus suitable, at least in theory, for soft evidence. IPFP's power lies in its ability to modify a joint distribution to satisfy *multiple* probability constraints through a simple iterative process. The virtual evidence method can easily be incorporated into the BN framework by adding a *virtual evidence node* to the network. In contrast, both Jeffrey's rule and IPFP update beliefs by manipulating the joint probability distributions, which is not directly supported by BN inference methods.

This paper reports our analysis of these three belief update methods and their interrelationships. We will show that when dealing with a single evidential finding, the belief update of both the virtual evidence method and Jeffrey's rule can be viewed as IPFP with a single constraint. Also, we present two methods we developed for belief update with multiple soft evidences and prove their correctness. Both of these methods integrate the virtual evidence method and IPFP, and they can be easily implemented as a wrapper on any existing BN inference engine.

3.1 Jeffrey's Rule and Soft Evidence

Consider a Bayesian network *N* over a set of variables *X* modeling a particular domain. *N* defines a joint distribution $P(X)$. When giving $Q(Y)$, an observation of a probability distribution on variables $Y \subseteq X$, Jeffrey's rule claims that the distribution of all other variables under this observation should be updated to

$$Q(X \setminus Y) = \sum_i P(X \setminus Y \mid y_i) Q(y_i), \quad \textbf{(3.1)}$$

where y_i is a state configuration of all variables in *Y*. Jeffrey's rule, also known as J-conditioning [36], assumes $Q(X \setminus Y \mid Y) = P(X \setminus Y \mid Y)$, i.e., invariance of the conditional probability of other variables, given *Y*, under the observation. Thus

$$\begin{aligned} Q(X) &= P(X \setminus Y \mid Y) Q(Y) \\ &= \frac{P(X \setminus Y, Y)}{P(Y)} Q(Y) \\ &= P(X) \frac{Q(Y)}{P(Y)} \end{aligned} \quad \textbf{(3.2)}$$

Here $Q(Y)$ is what we called soft evidence. Analogous to conventional conditional probability, we can also write $Q(Y)$ as $P(Y \mid se)$, where *se* denotes the soft evidence behind the soft evidential finding. *Y* is called the *evidence variables* of *se*. $P(Y \mid se)$ is interpreted as the posterior probability distribution of *Y* given soft evidence *se*. Then (3.1) and (3.2) can be rewritten as:

$$P(X \setminus Y \mid se) = \sum_{y_i} P(X \setminus Y \mid y_i) P(y_i \mid se) \quad \textbf{(3.3)}$$

and

$$P(X \mid se) = P(X)\frac{P(Y \mid se)}{P(Y)} \qquad \textbf{(3.4)}$$

The invariance of conditional probability of Jeffrey's rule can then be rewritten as $P(X \setminus Y \mid Y, se) = P(X \setminus Y \mid Y)$, which clearly indicates that evidence variables Y should d-separate soft evidence *se* from all other variables $X \setminus Y$.

Now consider an example from Pearl [35] for Jeffrey's rule:

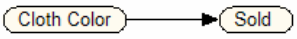


Figure 3.1 Example 3.1

Example 3.1. Suppose we are given a piece of cloth, which may be in one of three colors: blue, green, or violet, and may be sold the next day. The original joint distribution of the two variables is given as*:*

$$P(blue, sold) = 0.12, \quad P(blue, \overline{sold}) = 0.18,$$

$$P(green, sold) = 0.12, \; P(green, \overline{sold}) = 0.18,$$

$$P(violet, sold) = 0.32, \; P(violet, \overline{sold}) = 0.08.$$

Therefore the original belief about the color of the cloth (blue, green, violet) is (0.3, 0.3, 0.4). Now we observe the cloth by candle light, and find the color of the cloth has probability (0.25, 0.7, 0.05). By equation (3.2), this soft evidence leads to the posterior distribution

$$P(blue, sold \mid se) = 0.10, \quad P(blue, \overline{sold} \mid se) = 0.15,$$

$$P(green, sold \mid se) = 0.28, \; P(green, \overline{sold} \mid se) = 0.42,$$

$$P(violet, sold \mid se) = 0.04\,,\ \ P(violet, \overline{sold} \mid se) = 0.01\,.$$

Our belief on variable "color" is now updated from the original (0.3, 0.3, 0.4) to (0.25, 0.7, 0.05) in the posterior distribution, consistent with the soft evidence.

Pearl [36] has discussed the validity of the notation $P(A \mid se)$ under the assumption of Bayes' rule. Obviously, the original Bayes' rule does not agree with this notion since virtual evidence *se* is not an event. Also, although we can think that *se* must be caused by some events, those events are unknown to us (or even external to our model) and do not carry any prior or posterior distributions. Moreover, *se* is meaningless if it does not happen, e.g., $P(A \mid \overline{se})$ cannot be interpreted and calculated. However, its meaning is clear when used as the condition in a conditional probability. Similarly, we use *ve* for virtual evidence as the condition in a conditional probability. We use such notions in this paper because they are intuitive and it allows us to treat the soft evidence and virtual evidence as if they were events. If we agree that the prior distribution on *X* is a true understanding of the domain, then we should also agree that evidence (hard, virtual or soft) indicates an event, regardless of what form this event is, and that our belief (about some or all variables) should be updated when presented with evidence regardless its type, although the way the belief is to be updated may be different with different types of evidence.

3.2 Virtual Evidence

Besides Jeffrey's rule, virtual evidence method is also used in belief update when the uncertainty of observations of a variable's states is given in the form of a likelihood ratio.

A likelihood ratio represents the observer's strength of confidence toward the observed event. Suppose we are given variable A, which has states a_1, a_2, ..., a_n, its likelihood ratio $L(A)$ is defined as

$$L(A) = (P(Ob(a_1) \mid a_1) : P(Ob(a_2) \mid a_2) : \ldots : P(Ob(a_n) \mid a_n)),$$

where $Ob(a_i)$ denotes the event that we observed $A = a_i$ is *True* and $P(Ob(a_i) \mid a_i)$ is interpreted as the probability we observe A is in state a_i if A is indeed in state a_i.

A virtual evidential finding can also involve more than one variable. Let *ve* be a virtual evidence on $Y \subseteq X$ with a likelihood ratio $L(Y)$ (or $L(ve)$),

$$L(Y) = (P(Ob(y_1) \mid y_1) : \ldots : P(Ob(y_m) \mid y_m)),$$

where y_j is a state configuration of all variables in Y, and m is the total number of distinct configurations. The posterior probability of Y, given the evidence, is

$$\begin{aligned} P(Y \mid ve) &= c \cdot P(Y) \cdot L(Y) \\ &= c \cdot (P(y_1)L(y_1), \ldots, P(y_n)L(y_n)), \end{aligned}$$

(3.5)

where $c = 1/\sum_i P(y_i)L(y_i)$ is the normalization factor [36]. And since Y d-separates virtual evidence from all other variables, beliefs on $X \setminus Y$ are updated using Bayes' rule:

$$P(X \setminus Y \mid ve) = \sum_{y_i} P(X \setminus Y \mid y_i) P(y_i \mid ve) \qquad \textbf{(3.6)}$$

and similar to equation (3.2), this d-separation leads to

$$P(X \mid ve) = P(X)\frac{P(Y \mid ve)}{P(Y)} = P(X)\frac{c \cdot P(Y) \cdot L(Y)}{P(Y)} = c \cdot P(X) \cdot L(Y)$$

(3.7) Now we extend Example 3.1 to show how to use virtual evidence.

Example 3.2. Suppose we are not certain about our original belief about the cloth color. This information of uncertainty can be coded as virtual evidential findings: we are 50% confident that the cloth is blue when the cloth looks blue, 80% confident that the cloth is green when the cloth looks green, 100% confident that the cloth is violet when the cloth looks violet. Therefore our beliefs on the cloth color is updated using equation (3.5) as c·(0.3, 0.3, 0.4)·(0.5, 0.8, 1.0) = (0.19, 0.30, 0.51). Also by equation (3.7) we have the joint posterior distribution

$$P(blue, sold \mid ve) = 0.08, \quad P(blue, \overline{sold} \mid ve) = 0.11,$$

$$P(green, sold \mid ve) = 0.12, \quad P(green, \overline{sold} \mid ve) = 0.18,$$

$$P(violet, sold \mid ve) = 0.41, \quad P(violet, \overline{sold} \mid ve) = 0.10.$$

Our belief on "color" is updated by this virtual evidence from the (0.3, 0.3, 0.4) to (0.19, 0.30, 0.51).

Some BN inference engines support belief update with virtual evidence by directly taking the likelihood ratio as input. Otherwise, virtual evidence can be incorporated into any BN inference engine using a dummy node. This is done by adding a binary node ve_Y for the given $L(Y)$. This node, called virtual evidence dummy node by some, does not

have any child, and has all variables in Y as its parents. The conditional probability table (CPT) of ve_Y should conform with the likelihood ratio, i.e., for all i, j,

$$P(ve_Y \mid y_i) / P(ve_Y \mid y_j) = L(y_i) / L(y_j) .$$

Then, by instantiating ve_Y to *True*, the virtual evidence $L(Y)$ is entered into the BN and the belief can then be updated by any BN inference algorithm. In other words, a virtual evidence is equivalent to a hard evidence in an extended BN with the addition of a virtual evidence node.

3.3 IPFP on Bayesian Network

We can see that equations (3.4), (3.7) and (2.5) are in the same form. Therefore we can regard belief update with soft evidence by Jeffrey's rule as an IPFP process of a single constraint $P(Y \mid se)$, and similarly regard belief update with virtual evidence by likelihood ratio as an IPFP process of a single constraint $P(Y \mid ve)$. As such, we say that belief update by uncertain evidence amounts to change the given distribution so that 1) it is consistent with the evidence; and 2) it has the smallest I_1-divergence to the original distribution.

Moreover, IPFP provides a principal approach to belief update with multiple uncertain evidential findings. By treating these findings as constraints, the iterative process of IPFP leads to a distribution that is consistent with ALL uncertain evidences and is as close as possible to the original distribution.

Unlike the virtual evidence method, both Jeffrey's rule and IPFP cannot be directly applied to BNs because their operations are defined on the *full* joint probability distribution, and they do not respect the structure of BN [37]. For small BN, one can

explicitly generate the full joint distribution and then apply IPFP for belief update on the distribution. This is, however, infeasible for large BN, because, among other things, the distribution would be prohibitively large. Our proposed solutions to this problem are presented in the next section.

3.4 Inference with Multiple Soft Evidential Findings

Valtorta, Kim and Vomlel have devised a variation of Junction-Tree (JT) algorithm for belief update with multiple soft evidences using IPFP [40]. In this algorithm, when constructing the JT, all soft evidence nodes (i.e., those variables that are involved in any of the soft evidential findings) are fully connected with each other by additional edges. After triangulation, all soft evidence nodes appear in a clique (the *Big Clique*). Let C denote this big clique, $Y = \{Y_1, ..., Y_k\}$ and $\{se_1, ..., se_k\}$ denotes soft evidence variables and their respective soft evidences, and X denotes the set of all variables. This Big Clique algorithm first applies all hard evidences and updates $P(X)$ to $P^*(X)$ using traditional JT algorithm. Then, it absorbs soft evidences in C by updating the potential of C with the following IPFP formulae, iterating over $Q(Y_j)$:

$$Q_0(C) = P^*(C)$$
$$Q_i(C) = Q_{i-1}(C)\frac{P(Y_j \mid se_j)}{Q_{i-1}(Y_j)}$$

where $j = 1+(i\text{-}1) \bmod k$. The above procedure is iterated until $Q_n(Y_j)$ converges to $P(Y_j \mid se_j)$ for all j. Finally, $Q(C)$ is distributed to all other cliques, again using traditional JT algorithm.

This Big Clique algorithm is time efficient because it does not iterate on the joint distribution of all variables of the network but on the distribution of variables in a clique (which include all soft evidence nodes). It becomes inefficient in both time and space when the size of the big clique itself becomes large.

Besides the potential high cost of time and space, this Big Clique algorithm has another limitation. It works only with Junction-Tree, and thus cannot be adopted by those using other inference mechanisms[2]. Also, it requires incorporating IPFP operations into the JT procedure, causing re-coding of the existing inference algorithm. To address these issues, we propose two new algorithms for inference with multiple soft evidential findings. Both algorithms utilize IPFP, although in quite different ways. The first algorithm combines the idea of IPFP and the encoding of soft evidence by virtual evidence. The second algorithm is similar to the Big Clique algorithm but it *decouples* the IPFP with Junction-Tree. These two algorithms are presented in the next two subsections.

3.4.1 Iteration on the Network

As discussed in Section 3.2, inference with virtual evidence can be easily implemented using any BN inference methods with the help of a dummy node. With this dummy node, inference with virtual evidence (the likelihood ratio) is transformed to inference with hard evidence (instantiating the dummy node to true). This approach thus also works when multiple virtual evidential findings are present if we add dummy node for each finding.

[2] Valtorta and his colleagues also developed another algorithm for soft evidence inference, also based on JT inference engine [12]. This method does not require to form the big clique. Instead, it iteratively 1) updates the potential of the clique which contains variables of one soft evidence by (2.5) and 2) propagates the updated potential to the rest of the network. They mentioned the possibility of implementing this method as a wrapper around Hugin shell or other JT engines, but no suggestion of how this can be done was given [12].

As pointed out by Pearl [36], and Chan and Darwiche [5], soft evidence can be easily translated into virtual evidence when it is on a single variable. Given a soft evidence *se* on variable *A*, we want to find a likelihood ratio $L(A)$ such that

$$P(A) \cdot L(A) = P(A \mid se).$$

So

$$\begin{aligned} L(A) &= \frac{P(A \mid se)}{P(A)} \\ &= \left(\frac{P(a_1 \mid se)}{P(a_1)}, \frac{P(a_2 \mid se)}{P(a_2)}, \ldots, \frac{P(a_n \mid se)}{P(a_n)}\right). \end{aligned} \tag{3.8}$$

A soft evidence can also be defined on multiple variables, as illustrated by the following example.

Example 3.3. As depicted in Figure 2, we extend Example 3.1 by adding another variable *Cloth Pattern*, which also influences the salability of cloth in the next day.

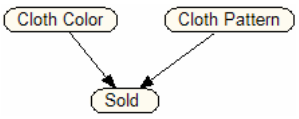


Figure 3.2 Example 3.3

The new observation can be made over both colors and patterns of the cloth, as a joint distribution below

$$Q(blue, Striped) = 0.12, \quad Q(blue, Solid) = 0.18,$$

$$Q(green, Striped) = 0.15, \; Q(green, Solid) = 0.15,$$

$$Q(violet, Striped) = 0.30, \; Q(violet, Solid) = 0.10.$$

To represent soft evidence on multiple variables, e.g., the one in Example 3.3 above, using virtual evidence, the likelihood ratio needs to be calculated from the joint distribution of soft evidence variables. Suppose we are given a soft evidence *se* on a set of variables $Y \subseteq X$, we can construct a virtual evidence *ve*, which, if applied, would have the same influence as *se* on variables in $X \setminus Y$. The likelihood ratio associated with this *ve* can be computed from the given evidential finding $P(Y \mid se) = Q(Y)$ and the original joint distribution $P(Y)$ in the way analogous to equation (3.8),

$$L(Y) = \left(\frac{P(y_1 \mid se)}{P(y_1)}, \ldots, \frac{P(y_m \mid se)}{P(y_m)}\right) \qquad \textbf{(3.9)}$$

where, again, y_i is a state configuration of all variables in Y, and m is the total number of distinct configurations. Then a dummy node can be created for this *ve* as described at the end of Section 3.2.

A problem arises when multiple soft evidences se_1, se_2, …, se_m are presented and converted to *ve* dummy nodes. Instantiating a single dummy node ve_i to *True* will have the same effect as applying the soft evidence se_i, in particular, the posterior probability of Y_i is made equal to $P(Y_i \mid se_i)$. This is no longer the case when all of these dummy *ve* nodes are set to *True*. Now, the belief on Y_i is not only influenced by ve_i, but also by all other dummy nodes which are working as hard evidences. As the result, the posterior probabilities of Y_i's are **NOT** equal to $P(Y_i \mid se_i)$. In other words, the soft evidences are not preserved or protected in the update process. It would be nice if we can hold the

probability of Y_i fixed after the corresponding ve_i is applied. Unfortunately, there is no such mechanism in BN.[3]

What is needed is a method that can convert a set of soft evidences to one or more likelihood ratios which, when applied to the BN, preserve every soft evidence $P(Y_i \mid se_i)$.

Algorithm 1 presented below accomplishes this purpose by combining the idea of IPFP and the virtual evidence method. Roughly speaking, this algorithm goes as follows. Like the IPFP, it is an iterative process and one soft evidence se_i is considered at each iteration. If the current probability of Y_i equals $P(Y_i \mid se_i)$, then it does nothing, otherwise, a new virtual evidence is created based on the current probability of Y_i and the evidence $P(Y_i \mid se_i)$. We will show that this process converges, and when it converges, the probability of Y_i is equal to $P(Y_i \mid se_i)$. To better describe the algorithm, we adopt the following notations:

- P (without subscript): the prior probability distribution.
- P_k (with subscript): the probability distribution at k^{th} iteration.
- $ve_{i,j}$: the j^{th} virtual evidence created for the i^{th} soft evidence.

Algorithm 3.1. Consider a BN N with prior distribution $P(X)$, and a set of m soft evidential findings $SE = (se_1, se_2, \ldots, se_m)$ with $P(Y_1 \mid se_1),\ldots, P(Y_m \mid se_m)$. We use the following iteration method for belief update:

[3] This would not be a problem for hard evidence because hard evidence is protected with special treatment of probabilities of zero. For example, in Junction-Tree algorithm, the potential algebra defines that zeroes in probability tables remain zeros after each operation [15]. This would not be a problem for virtual evidence, either, because virtual evidence (the likelihood ratio) is protected in the CPT of the dummy node, which is never changed during inference.

function Soft_Evidential_Update(N)

1 $P_0(X) = P(X)$; $k = 1$;

2 Repeat the following until convergence:

3 $i = 1 + (k-1) \bmod m$; $j = 1 + \lfloor (k-1)/m \rfloor$;

4 Construct virtual evidence $ve_{i,j}$ with likelihood ratio

$$L(Y_i) = \left(\frac{P(y_{i,1} \mid se)}{P_{k-1}(y_{i,1})}, \ldots, \frac{P(y_{i,s} \mid se)}{P_{k-1}(y_{i,s})}\right),$$

where $y_{i,1}, \ldots, y_{i,s}$ are state configurations of Y_i;

5 Obtain $P_k(X)$ by updating $P_{k-1}(X)$ with $ve_{i,j}$ using standard BN inference;

6 $k = k + 1$;

The algorithm cycles through all soft evidences in SE. At the k^{th} iteration, the i^{th} soft evidence se_i is selected (step 3) to update the current distribution $P_{k-1}(X)$. This is done by constructing a virtual evidence $ve_{i,j}$ according to equation (3.9). The second subscript here, j, is the number of virtual evidences created for se_i, this index is incremented in every m iterations. When converged, we can form a single virtual evidence node ve_i for each soft evidence se_i with the likelihood ratio that is the product of likelihood ratios of all $ve_{i,j}$, $ve_i = \prod_j ve_{i,j}$.

The convergence and correctness of Algorithm 1 is established in Theorem 3.1 below.

Theorem 3.1. If the set of soft evidence $SE = (se_1, se_2, \ldots, se_m)$ is consistent, then Algorithm 1 converges with joint distribution $P^*(X)$, in which $P^*(Y_i) = P(Y_i \mid se_i)$ for all se_i in SE.

Proof.

By equations (3.8) and (3.9), the likelihood ratio computed at Step 4 satisfies

$$P_{k-1}(Y_i) \cdot L(Y_i) = P(Y_i \mid se_i) .$$

To see what is achieved at Step 5, re-write equation (3.7),

$$\begin{aligned} P_k(X \mid ve) &= c \cdot P_{k-1}(X) \cdot L(Y_i) \\ &= P_{k-1}(X) \frac{P(Y_i \mid ve)}{P_{k-1}(Y_i)} . \end{aligned}$$

This is the same as equation (2.5). Therefore, Algorithm 3.1 performs IPFP on $P_0(X)$ with soft evidences $P(Y_1 \mid se_1), \ldots, P(Y_m \mid se_m)$ as constraints. Then following Theorem 2.1, Algorithm 3.1 converges and all constraints (i.e., soft evidences) are satisfied when it converges.

3.4.2 Iteration on Local Distribution

Algorithm 3.1 may become expensive when the given BN is large because it updates the beliefs of the entire BN in each iteration (step 5). Following is another algorithm that iterates virtual evidence on joint distribution of only evidence variables:

Algorithm 3.2. Consider a Bayesian network N and a set of m soft evidential findings $SE = (se_1, se_2, \ldots, se_m)$ to N with $P(Y_1 \mid se_1), \ldots, P(Y_m \mid se_m)$. Let $Y = Y_1 \cup \ldots \cup Y_m$. We use the following iteration method for belief update in N:

function Soft_Evidential_Update(N)

1 Use any BN inference method on N to obtain $P(Y)$;

2 Apply IPFP on $P(Y)$, using $P(Y_1 | se_1)$, $P(Y_2 | se_2)$, …, $P(Y_m | se_m)$ as the probability constraints and obtain $P(Y | se_1, se_2, \ldots, se_m)$;

3 Add a virtual evidence dummy node to N to represent $P(Y | se_1, se_2, \ldots, se_m)$ with likelihood ratio $L(Y)$ calculated according to equation (3.9);

4 Apply $L(Y)$ as a single virtual evidence to update beliefs in N.

Algorithm 3.2 also converges to the I_1-projection of $P(X)$ on the set of soft evidences SE, even though the iterations are carried out only on a subset of X.

Theorem 3.2. Let $R_1(Y_1)$, $R_2(Y_2)$, …, $R_m(Y_m)$ be probability constraints on distribution $P(X)$. Let $Y = \bigcup_i Y_i$ and $Y \subseteq Z \subseteq X$. Suppose from IPFP we get the I_1-projection of $P(Y)$ on $\{R_1, R_2, \ldots, R_m\}$ as $Q(Y)$ and the I_1-projection of $P(Z)$ on $\{R_1, R_2, \ldots, R_m\}$ as $Q'(Z)$. Let $Q(X)$ and $Q'(X)$ be obtained by applying the Jeffrey's rule on $P(X)$ using $Q(Y)$ and $Q'(Z)$. Then $Q(X) = Q'(X)$.

Proof. From the definition of I-divergence we have

$$I(Q'(Z) \| P(Z))$$
$$= \sum_z Q'(z) \log \frac{Q'(z)}{P(z)}$$

$$= \sum_z Q'(z \setminus y \mid y) Q'(y) \log \frac{Q'(z \setminus y \mid y) Q'(y)}{P(z \setminus y \mid y) P(y)}$$

$$= \sum_z Q'(z \setminus y \mid y) Q'(y) \log \frac{P(z \setminus y \mid y) Q'(y)}{P(z \setminus y \mid y) P(y)}$$

$$= \sum_z Q'(z \setminus y \mid y) Q'(y) \log \frac{Q'(y)}{P(y)}$$

$$= \sum_y (\sum_{z \setminus y} Q'(z \setminus y \mid y)) Q'(y) \log \frac{Q'(y)}{P(y)}$$

$$= \sum_y Q'(y) \log \frac{Q'(y)}{P(y)}$$

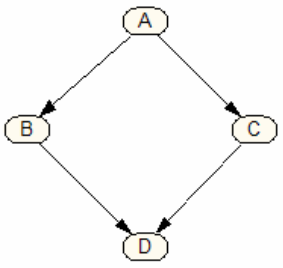
$$= I(Q'(Y) \,\|\, P(Y)) \qquad \textbf{(3.10)}$$

Note that line 4 comes from the fact that $Q'(z \setminus y \mid y) = P(z \setminus y \mid y)$, the invariance of conditional probability, as discussed in Section 3.1. Also, note that line 7 comes from the fact that $\sum_{z \setminus y} Q'(z \setminus y \mid y) = 1$.

Note the IPFP that generates I_1-projection of $P(Z)$ on $\{R_1, R_2, \ldots, R_m\}$ minimizes $I(Q'(Z)||P(Z))$, then by equation (3.10) it also minimizes $I(Q'(Y)||P(Y))$. Also because the IPFP that generates the I_1-projection of $P(Y)$ on $\{R_1, R_2, \ldots, R_m\}$ minimizes $I(Q(Y)||P(Y))$, by the uniqueness of I_1-projection we have $Q'(Y) = Q(Y)$. Then from Jeffrey's rule we know $Q(X) = Q'(X)$.

Example 3.4. As depicted in Figure 3.3, we are given a Bayesian network with variables *A, B, C,* and *D* and two soft evidences $P(B) = (0.7, 0.3)$ and $P(C) = (0.3, 0.7)$. Figure 3.4(a) shows the running result of Algorithm 3.1, with resulting likelihood ratios $L(B) = (1.0, 0.354)$ and $L(C) = (0.578, 1.0)$ at convergence. Figure 3.4(b) shows the running

result of Algorithm 3.2, where $L(B, C)$ = (0.578, 1.0, 0.205, 0.354) at convergence. Both algorithms converge in 4 iterations to the same distribution.



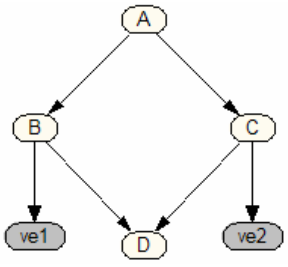
A	
1	**0**
0.4	0.6

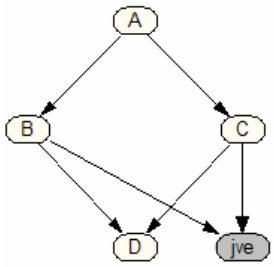
A	B	
	1	**0**
1	0.20	0.80
0	0.60	0.40

A	C	
	1	**0**
1	0.60	0.40
0	0.35	0.65

B	C	D	
		1	**0**
1	**1**	0.10	0.90
1	**0**	0.85	0.15
0	**1**	0.45	0.55
0	**0**	0.70	0.30

Figure 3.3 The BN of Example 3.4





(a) The running result of Alg 3.1 **(b) The running result of Alg 3.2**

Figure 3.4 Running result of Example 3.4

3.4.3 Time and Space Performance

The iterations of Algorithm 3.1, Algorithm 3.2 and the Big Clique algorithm all lead to the same distribution. However, at each iteration, the Big Clique algorithm updates

beliefs of the joint probabilities of the big clique C, Algorithm 3.2 updates the belief of evidence variables Y, and Algorithm 3.1 updates the belief of the whole BN, or, of all variables in X. Clearly, $Y \subseteq C \subseteq X$. However, the time complexity for one iteration of Big Clique is exponential to $d(C)$ (the state number of C), and Algorithm 3.2 exponential to $d(Y)$, because both require modifying a joint distribution (or potential) table. On the other hand, the time complexity of Algorithm 3.1 equals to the complexity of the BN inference algorithm it uses for belief update, e.g., if we use JT, the time complexity for one iteration of Algorithm 3.1 is exponential to the size of the largest clique in JT, which may be smaller than C and Y, especially for small and sparse BNs.

Both the Big Clique and Algorithm 3.2 are space inefficient. The Big Clique needs additional space for the joint potential of C, whose size is exponential to $d(C)$. Algorithm 3.2 also needs additional space for the joint distribution of Y, and the dummy node of virtual evidence in Step 4 leads to a CPT with size exponential to $d(Y)$. In contrast, Algorithm 3.1 only needs additional space for virtual evidence, which is linear to $d(Y)$.

Algorithm 3.2 is thus more suitable for problems with a large BN but a few soft evidential findings and Algorithm 3.1 is more suitable for small to moderate-sized BNs. Also, both Algorithm 3.1 and 3.2 have the advantage that users do not have to stick to and modify the Junction-Tree when conducting inference with soft evidence. They can be easily implemented as wrappers on any BN inference engine.

3.5 Experiments and Evaluation

To empirically evaluate our algorithms and to get a sense of how expensive these approaches may be, we have conducted two experiments with artificially made networks

of different sizes. Our two algorithms are implemented in Java using Netica[4] Java API and its JT based inference engine. The reported memory consumption does not include those that were used by the Junction-Tree, but the reported running time is the total running time.

The first experiment used a BN of 15 binary variables. Three sets of 2, 4, 8 soft evidential findings were selected for the experiments. One half of these evidential findings involved 2 variables, and the other half involved 1 variable. In the implementation of Algorithm 3.2 we used I-divergence to measure the distance of the probability distributions of two iterations. In the implementation of Algorithm 1, because it is very time consuming to calculate the I-divergence between two probability distributions of a large number of variables, we compute the cross-entropy of every variable and sum them up. Because of this difference, the Algorithm 3.1 runs more iterations than Algorithm 3.2 to ensure the result is sufficiently accurate. The experiment results are given in Table 3.1.

From Table 3.1 we can see that both the time and memory consumptions of Algorithm 3.1 increase slightly when the number of evidences increases. However, those for Algorithm 3.2 increase rapidly, consistent with our analysis.

Table 3.1 Experiment 1

# of findings	**# Iterations (Alg 3.1|Alg 3.2)**		**Exec. Time (Alg 3.1|Alg 3.2)**		**Memory (Alg 3.1|Alg 3.2)**	
2	24	14	0.57s	0.62s	590,736	468,532
4	79	23	0.63s	0.83s	726,896	696,960
8	95	17	0.71s	15.34s	926,896	2544,536

[4] Netica: Bayesian network tool from Norsys Software Corp. http://www.norsys.com/

The second experiment used BNs of 30, 60, 120, and 240 binary variables. In all cases we entered the same 4 soft evidential findings involving a total of 6 variables. All 4 experiments converge after the same number of iterations (43 for Algorithm 1 and 14 for Algorithm 2). From Table 3.2 we can see that when the number of soft evidences is fixed, the running time of Algorithm 3.2 increases slightly with the increase of the network size. Especially, the time for IPFP (the time in parentheses) keeps stable when the variable number increases, which means that most increased time was spent on constructing the joint probability distribution from the BN (Step 1 of Algorithm 3.2). These experiment results confirm our theoretical analysis for the proposed algorithms.

Table 3.2 Experiment 2.

Size of *N*	**# Iterations (Alg 1|Alg 2)**		**Exec. Time (Alg 3.1|Alg 3.2 (IPFP))**		**Memory (Alg 3.1|Alg 3.2)**	
30	43	14	0.58s	0.67s (0.64s)	721,848	691,042
60			0.71s	0.69s (0.66s)	723,944	691,424
120			1.71s	0.72s (0.66s)	726,904	691,416
240			103.1s	3.13s(0.72s)	726,800	696,842

3.6 Summary

In this chapter, we analyzed three existing belief update methods for Bayesian networks: virtual evidence, Jeffrey's rule and IPFP. We established that belief update with one virtual evidence or soft evidence is equivalent to an IPFP with a single constraint. Besides, IPFP can be easily applied to BN with the help of virtual evidence. Our proposed algorithms update probability beliefs for multiple soft evidences by integrating methods of virtual evidence, IPFP and traditional BN inference with hard evidence. Compared with previous soft evidential update methods such as the Big Clique, our

algorithms have practical advantage of being independent of any particular BN inference engine.

4 Variable Linkage

In this Chapter, we consider the problem of how to model variables in SLBN that share similar semantics. First we present our thoughts about what type of semantic similarity is considered as shareable between variables. Base on this understanding, we propose the variable linkage to represent semantic similarity and give detailed examples of different use cases of variable linkages. At the end of this chapter we present an assumption we make on the variable linkages of SLBN.

4.1 Semantic Similarity

In our proposed SLBN, Bayesian networks are trying to communicate and exchange beliefs via similar concepts, where a single concept is represented by one of more variables. Certainly, a method that quantifies similarities is needed for the framework. However, "similarity" is not a well-defined property for concepts, and neither is its quantification. It is difficult to represent what we mean when we say "concepts *A* and *B* are similar", even in an intuitive way. Our natural language has a very vague definition for the word "similar". Here is how dictionaries explain the word "similar":

1. *having characteristics in common*
2. *alike in substance or essentials*
3. *not differing in shape but only in size or position*

——www.merrian-webster.com

1. *marked by correspondence or resemblance*
2. *having the same characteristics*
3. *(of words) expressing closely related meanings*
4. *capable of replacing or changing places with something else; permitting mutual substitution without loss of function or suitability*

——WordNet

These straight forward and intuitive definitions are not accurate and sufficient for describing similarities between complex concepts or concepts with specific properties in intelligence systems. For example, the concepts *High-tech Company Employee* and *High-income People* are considered as similar because most high-tech company employees are high-income people as well as a large portion of high-income people work for high-tech companies. And the more high-tech company employees are classified as high-income people, the more the concept *High-tech Company Employee* is similar to the concept *High-income People*. However, it is hard to articulate these two concepts have "characteristics in common" or are "alike in substance or essentials". Another example is to think about two concepts *Computer Keyboard* and *Typewriter*, which by definition from the dictionaries can be said as similar because they have some common characteristics, such as both of them contains keys of characters and numbers with the same layout. But no instances can be both a computer keyboard and a typewriter. So although such similarity can be expressed, it cannot be quantified and used to relate the status of the two concepts. **So our interpretation to similarity is that t*wo concepts are semantically similar as their common instances have a certain portion of share.***

Moreover, “similar” is not a symmetric relation between concepts in our concern. Rather than stating concepts *A* and *B* are similar, we state *A* is similar to *B* and/or *B* is similar to *A*. We emphasize the direction because the domain knowledge may only capture and quantify the similarities of one direction. Let’s think about the concepts *High-tech Company Employee* and *High-income People* again: how likely a high-tech company employee receives high income and how likely a high-income person works in a high-tech company are two different estimations. Since domain knowledge may only know the information of one direction, *similarities are quantified and utilized with direction*.

The last issue is how to quantify the semantic similarities. Two concepts are identical if their instances are all shared by each other. Two concepts are dissimilar if they share no common instances. Also, as concept *A* has more instanced shared with the other concept *B*, *A* is more similar to *B*, e.g., *A* and *C* each has 100 instances, *A* share with *B* 80 instances, and *C* share with *B* 50 instances; then we say *A* is more similar to *B* than *C* is. So *the similarity can be represented by the ratio of the shared instances to all the instances*.

Following the above statements, we find that *conditional probability is a natural representation of semantic similarities*. For example, *P*(*High-tech Company Employee* | *High-income People*) can be interpreted as a measurement of how likely a high-income person works in a high-tech company.

4.2 Variable Linkage

Given two separately developed BNs N_A and N_B, the precondition for conducting reasoning across them is that there exists an overlap in the domains they model. This overlap in domain could be represented in different forms. For example, in MSBN the overlap knowledge is represented as a Junction-Tree of shared variables, in AEBN this overlap is represented as shared variables, and such overlap knowledge could also be in the form of a distribution, a function, or a set of logic rules. In our framework, we require this overlap of knowledge overlap to be encoded as conditional probability between variables in Bayesian networks.

4.2.1 Pair-Wise Variable Linkage

Definition 4.1 (*Pair-wise Variable Linkage*): A pair-wise variable linkage represents the semantic similarity between two variables in separate BNs. A pair-wise variable linkage L_B^A from variable A in Bayesian network N_A to variable B in Bayesian network N_B is defined as

$$< A, B, N_A, N_B, S_B^A >,$$

where S_B^A is the quantification of the semantic similarity that L_B^A represents. S_B^A is a in the form of an $m \times n$ matrix:

$$S_B^A(i, j) = \{P(b_j | a_i)\},$$

where $i = 1, \ldots, m$, $j = 1, \ldots, n$, m and n are the number of the states of of A and B, respectively.

We call variables A as L's *source variable* or *source*, and B as L's *destination variable* or *destination*. The source variables of linkage L is denoted by *src*(L), and the destination variables of linkage L is denoted by *dest*(L). Also N_A is called the *source BN* of L and BN_B the *destination BN*. In the rest of this dissertation, S_B^A is always referred as a matrix. If we want to involve S_B^A in the calculations defined in the probability distribution algebra, S_B^A is presented in the form of conditional probabilities.

Although semantic similarity is quantified by conditional probabilities in variable linkage, it is between concepts that are inherently semantically similar, not those that have general probabilistic dependencies. The essential difference is that probabilistic dependency may not remain unchanged when some events occur in N_A and N_B. Interdependencies encoded by the DAG of Bayesian networks are such relations that depend on the status of variables. In contrast, semantic similarity persists regardless of occurrences of other events.

Example 4.1 (Probabilistic Dependency vs. Semantic Similarity): In two medical diagnosis BN N_A and N_B, we have variables *Diabetes* in N_A and *Hyperglycemia* in N_B, both of which has states *True* and *False*. Their relation can be captured by a conditional probability

$$P(Hyperglycemia \mid Diabetes) = \begin{pmatrix} 1 & 0 \\ 0.2 & 0.8 \end{pmatrix}$$

as diabetes is characterized by variable hyperglycemia. This conditional probability embodies a conditional dependency in that hyperglycemia is the effect of diabetes, and can be influenced by the states other variables take, for example, *Obesity*. And now if in

N_A there is another variable *Blood Sugar Level*, which has three state *High*, *Medium*, and *Low*. Then its relation between variable *Hyperglycemia* can be captured by the following conditional probability:

$$P(Hyperglycemia \mid Blood\ Sugar\ Level) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

This conditional probability represents the semantic similarity between two variables and it will not change upon the occurrences of any events.

Besides the above distinction, the probabilistic influences on variable linkages also differ from the probabilistic influences on BN edges in that they can only be propagated along the direction of the linkages as the semantic similarity is directed. Since variable linkages do not represent conditional dependencies, Bayes rule cannot be applied directly for inference. Although a linkage is quantified by $P(B \mid A)$, the reverse similarity $P(B \mid A)$ is not known to the linkage, and it can not be obtained by Bayes rule. Suppose we are given a linkage L_B^A = < *A*, *B*, N_A, N_B, S_B^A >, where $P(B \mid A)$ is used to quantify how *B* is similar to *A*. If we calculate the conditional probability of the reverse side by Bayes rule, then we should have

$$P(A \mid B) = P(B \mid A)\frac{P(A)}{P(B)},$$

where $P(B \mid A)$ is invariant with respect to any events. We can see that this $P(A \mid B)$ may be dependent to the prior belief on *A* and *B*. Therefore, variable linkages do not support the probabilistic influence of the reverse direction. Variable linkages are bridges

by which probabilistic influence can be propagated and cannot be simply reversed without additional domain knowledge. In SLBN, the belief change of source variables influences the belief on destination variables and the belief change of destination variables indicates the belief on source variables has been changed.

Moreover, it is enough that the conditional probabilities of a semantic similarity retains only under the events within the current domain. That is to say, there could exist some event outside the domain of the linked BNs that invalids the variable linkage, but as long as such event is not discussed, the variable linkage could be established. For example, the equivalence of variables *Male Parent* and *Father* are absolutely identical. Even in the real world, the conditional probabilities describing their equivalence cannot be changed by any event. However, this is not the case in many situations. In the following example, we will show that a variable linkage is invalided by an external event, but is valid for describing a specific problem:

Example 4.2: Suppose we are given two similar variables *Precipitation* in N_A and *Raining* in N_B, both of which has states *True* and *False*. And upon our observation, a variable linkage is established from *Precipitation* to *Raining*, and the linkage is quantified as follows:

$$P(Raining \mid Precipitation) = \begin{pmatrix} 0.88 & 0.12 \\ 0 & 1 \end{pmatrix}.$$

Suppose N_A describes a meteorology model and N_B models how weather interacts with human's activity, and then this variable linkage could help to discover more relations between human's activities and meteorology phenomenons. Now that somebody proposes a variable *Season*, which gives new visions about the above problem but is not

modeled by either N_A or N_B. If variable *season* is considered, the above variable linkage is invalided because the relation of *Precipitation* and *Raining* is different in different seasons. Specifically,

$$P(Raining \mid Precipitation, Season) \neq P(Raining \mid Precipitation).$$

However, as long as the variable *Season* is not included in the domain, the conditional probability for the variable linkage from *Precipitation* to *Raining* is still invariant w.r.t. the events in the linked BNs.

4.2.2 Variable Linkage

Semantic similarity is not restricted to only one to one mappings, it can also be between multiple variables. For example, suppose concept *A* is similar to the union of *B* and *C*, then two pair-wise linkages from *A* to *B* and *A* to *C* would not be enough to describe such a similarity. So next we will give a general definition for variable linkage, which is between two sets of variables. However, we have to add restrictions to the set of linked variables so that inference can be conducted properly across the linkages. We believe that these restrictions are reasonable and can be easily met by many Bayesian networks.

Definition 4.2 (*variable linkage*): A variable linkage represents the semantic similarity between two sets of variables in separate Bayesian networks. A variable linkage L_Y^X from a set of variables X in Bayesian network N_X to a set of variable Y in Bayesian network N_Y is defined as

$$< X, Y, N_X, N_Y, S_Y^X >,$$

where S_Y^X is the quantification of the semantic similarity that L_Y^X represents. $\forall X_i, X_j \in X$ and $Y_i, Y_j \in Y$, one of the following conditions must be satisfied:

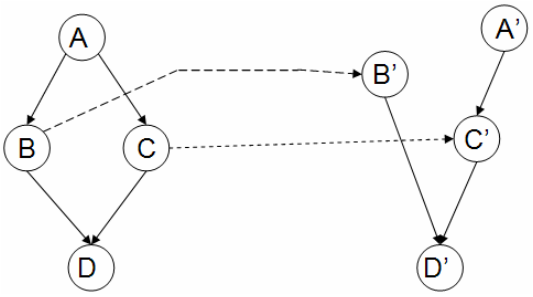
1. There is no directed path from X_i to X_j and from Y_i to Y_j;
2. If there is a directed path from X_i to X_j or from Y_i to Y_j, then any variable in *path*(X_i, X_j) or *path*(Y_i, Y_j) is also in X or Y.

S_Y^X is a conditional probability distribution in the form of an $m \times n$ matrix:

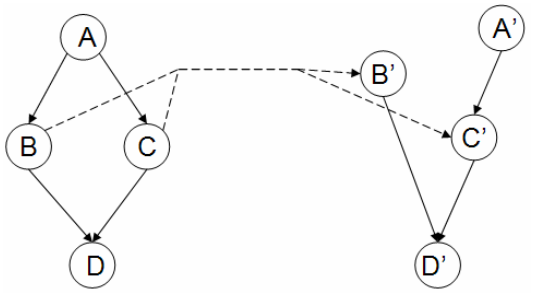
$$S_Y^X(i, j) = \{P_S(y_j \mid x_i)\},$$

where $i = 1, \ldots, m$, $j = 1, \ldots, n$, m and n are the number of the state configurations of X and Y, respectively.

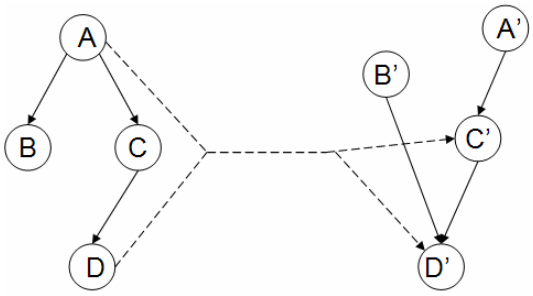
Considering the semantics of Bayesian networks, variable linkage's restrictions about the linked variables are quite reasonable. If a variable linkage does not obey these constraints, then there must exist a variable A in the networks such that one of its ancestors and one of its descendants are included in a linkage. Let B denotes the ancestor and C denote the descendant, then we have *path*(B, A, C). The linkage that involves B and C indicates that both B and C are similar to a concept in another BN, but variable A, which is an effect of B and a cause of C, is not similar to that concept. This is not quite reasonable.



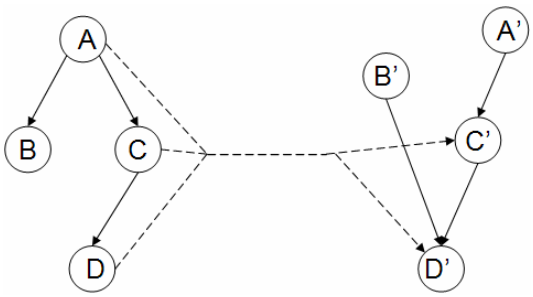
(a) pair-wise variable linkage



(b) variable linkage satisfying condition 1



(c) Invalid variable linkage according to condition 2



(d) variable linkage satisfying condition 2

Figure 4.1 Variable linkages

Figure 4.1 illustrate our definition of variable linkages. Figure 4.1(a) depicts two pair-wise variable linkages $L^{B}_{B'}$ and $L^{C}_{C'}$. Figure 4.1(b) depicts a variable linkage $L^{BC}_{B'C'}$, which meets the Condition 1 of the variable linkage's definition. Figure 4.1(c) is not a valid variable linkage because according to Condition 2 of the variable linkage's definition, variable *C* should also be included in the linkage. And Figure 4.1(d) shows the correction.

4.2.3 Expressiveness of Variable Linkage

Based on our definition, the relations that can be represented by the variable linkages can be divided into the following categories:

1. Logical relationships defined in OWL syntax, including equivalent, union, intersection, and subclass complement. These relations can be represented by variable linkages in both directions. The conditional probabilities can be determined as what were done in BayesOWL [11][12] while most entries can be determined logically and a few need domain knowledge.

2. Relaxation of logical relationships by replacing set inclusion by overlapping. For example, two concepts *A* and *B* are similar is a relaxation of equivalence, where *A* and *B* include each other is replaced by *A* and *B* overlap with each other.

3. Equivalence relations but same concepts are modeled as different variables (e.g., *Hyperglycemia* and *Blood Sugar Level*, they not only have different names but also have different number of states). This type can be easily treated as extensions to either 1 or 2 above.

In the above statements, you may find that the semantics of the variable linkage is different from what the semantic similarity expected in Section 4.1. For example, a variable linkage can represent the relationship of complementation, such as *Man* and *Woman*, but these concepts do not share instances and cannot be considered as semantically similar. This is because SLBN represents semantic similarities in propositional logic level. In BN, a concept is represented by variables with states, each of which can be interpreted as a proposition. For example, the variable *Man* has two states *True* and *False*, which stand for two propositions *Man = True* and *Man = False*. Each of these propositions is supported by a set of instances. Therefore, in SLBN, ***a variable linkage represents semantic similarities between propositions, rather than between variables***. When we say concept *High-tech Company Employee* is similar to the concept *High-income People*, we mean the proposition *High-tech Company Employee = True* is similar to the proposition *High-income People = True*. So in SLBN, a variable linkage between variables *Man* and *Woman* actually represents semantically similar propositions *Man = True* and *Woman = False*.

Illustrative examples of the above categories are given below.

Example 4.3 (Category 1):

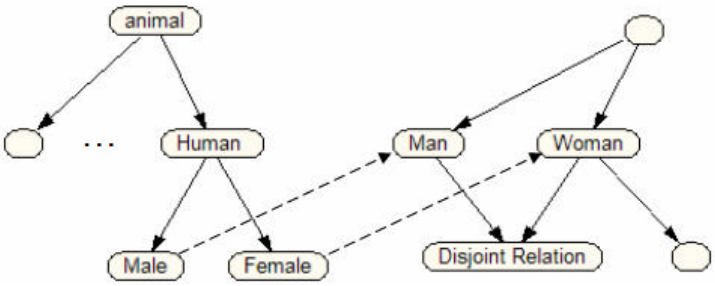


Figure 4.2 Variable linkages for identical variables

Figure 4.2 depicts an example of variable linkages for equivalent variables. Here variables *Male*, *Female* are sub-concepts of *Human*. They and *Man* and *Woman* all have two *states* true and *false*. And the linkages are quantified as

$$S_{Man}^{Male} = S_{Woman}^{Female} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Example 4.4 (Category 1):

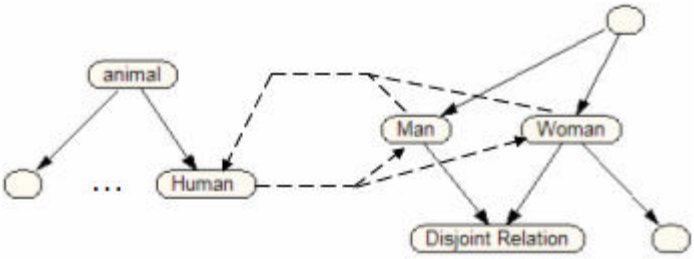


Figure 4.3 Variable linkages maps one variable with the union of two variables

Figure 4.3 depicts an example of two variable linkages of opposite direction. Here the *Human* is a concept identical to the union of *Man* and *Woman*. The linkage $L_{Man,Woman}^{Human}$ is quantified as

$$S_{Man,Woman}^{Human} = \begin{pmatrix} 0 & 0.51 & 0.49 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and the linkage $L_{Human}^{Man,Woman}$ is quantified as

$$S_{Human}^{Man,Woman} = \begin{pmatrix} - & - \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Example 4.5 (Category 2):

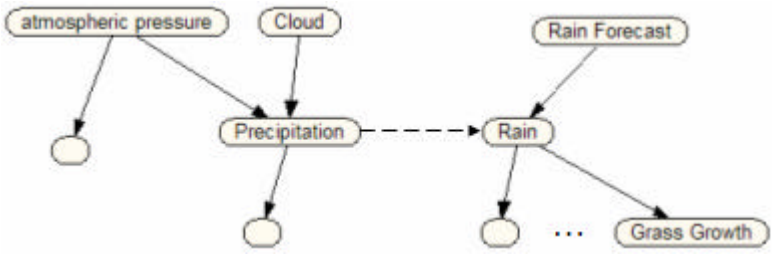


Figure 4.4 Variable linkages from a concept to its sub-concept

Figure 4.4 depicts a variable linkage from a super concept *Precipitation* to a sub concept *Rain*, both of which has two states *True* and *False*. The linkage is quantified as

$$S_{Rain}^{Precipitation} = \begin{pmatrix} 0.85 & 0.15 \\ 0 & 1 \end{pmatrix}.$$

A linkage defined from a sub-concept A to a super-concept B should be carefully treated. Regularly, $P(B \mid \overline{A})$ cannot be well assessed as $\overline{A}$ is unknown or will be changed by the occurrence of events. To safely define a linkage from a sub-concept to a super-concept, (B - A) should be defined using some other variables and the linkage should be

created from the union of *A* and (*B* - *A*) to *B*. The next example shows how to define a linkage in such situation.

Example 4.6 (Category 2):

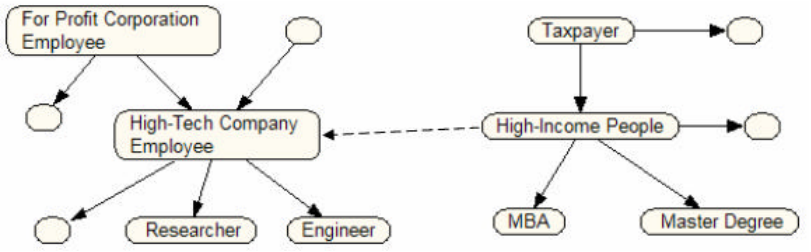


Figure 4.5 A variable linkage with context specific similarity

Figure 4.5 depicts a variable linkage for Example 4.1. Suppose both linked BNs are constructed using statistics data of the same area. *High-tech Company Employee* and *High-Income People* are two concepts overlap in domain knowledge, and both concepts are represented by variables with states *True* and *False*. The linkage L_{HTCE}^{HIP} is quantified as

$$S_{HTCE}^{HIP} = \begin{pmatrix} 0.7 & 0.3 \\ 0.01 & 0.99 \end{pmatrix}.$$

Example 4.7 (Category 3):

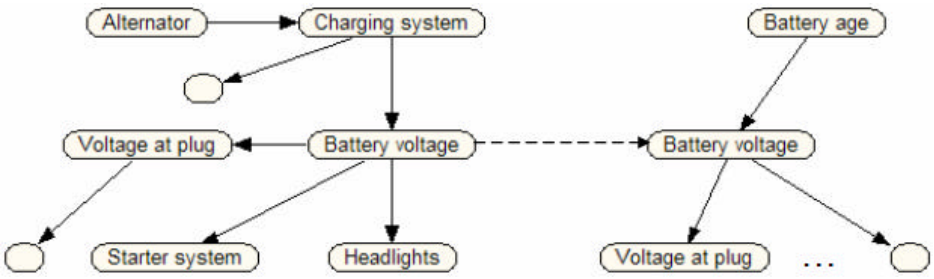


Figure 4.6 A variable linkage for identical variables with different descriptions

Figure 4.6 depicts an example of a variable linkage for identical concepts represented by different variables. The source variable *Battery voltage* has two states *Strong* and *Weak*, where the destination variable *Battery voltage* has three states *Strong*, *Weak* and *Dead*. Then the linkage is quantified as

$$S^{Battery\_voltage}_{Battery\_voltage} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.9 & 0.1 \end{pmatrix}.$$

4.3 Consistency between Variable Linkages and Linked Bayesian Networks

When a variable linkage is created to represent semantic similarities between variables, then what is the relation between the similarity and the beliefs of the linked BNs? Suppose we are given L^A_B = < *A*, *B*, N_A, N_B, S^A_B >, then the prior distribution of the linked variables should conform the similarity, e.g., if *A* is identical to *B*, then *P*(*A*) in N_A equals *P*(*B*) in N_B. To formally state this, we introduce the concept of *consistency* of linked BNs and their variable linkages.

Definition 4.3 (*Consistency of Variable Linkage*): A variable linkage L^X_Y = <*X*, *Y*, N_X, N_Y, S^X_Y > and its linked BNs N_X and N_Y are consistent if and only if the probability distribution of the linked variables can be represented by a single joint probability distribution. Specifically, the prior distributions of linked variables must satisfy the linkage's similarity:

$P_Y(Y) = S^X_Y P_X(X)$ in the matrix notation, or $P_Y(y_i) = \sum_j P_S(y_i \mid x_j) P_X(x_j)$.

And there exists a joint probability distribution $Q(X, Y, p(X), p(Y))$ such that the following constraints are satisfied:

1. $Q(X,Y) = P_S(Y \mid X)P_X(X) = S_Y^X \cdot P_X(X)$;
2. $Q(X,p(X)) = P_X(X,p(X))$ and $Q(Y,p(Y)) = P_Y(Y,p(Y))$;
3. $Q(p(X),p(Y)) = f(p(X),p(Y))$;

where $p(X)$ represents X 's parents and $p(Y)$ represents the Y's parents who are not involved in any other linkages. A simple yet reasonable f would be that $p(X)$ and $p(Y)$ are independent of each other, i.e., $f(p(X), p(Y)) = P(p(X))P(p(Y))$. A set of linked BNs are said to be consistent if all linkages and their linked BNs are consistent.

If a variable linkage and its linked BNs are consistent, then we can construct one of such joint distribution Q by IPFP on an initially uniform distribution. First we initiate a probability distribution as

$$Q_0(X,Y,p(X),p(Y)) = \left\{ \frac{1}{d(X)d(y)d(p(X))d(p(Y))} \right\}.$$

Then we run IPFP on $Q_0(X, Y, p(X), p(Y))$ with the constraints: $Q(X, Y)$, $Q(X, p(X))$, $Q(Y, p(Y))$, and $Q(p(X), p(Y))$ as given in Definition 4.3. Conversely, if the IPFP does not converge, then we can conclude that the given variable linkage and its linked BN are not consistent.

Figure 4.7 depicts an example of consistent linked BNs. Figure 4.7(a) shows two linked BNs with a linkage between two disjoint variables. The prior distributions of B and D are respectively (0.76, 0.24) and (0.24, 0.76), which already represent they are

disjoint concepts. Figure 4.7(b) shows the joint probability distribution $Q(A, C, B, D)$, which satisfies the following constraints:

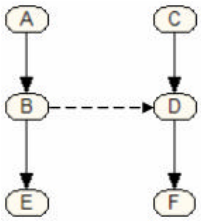
$Q(A, C) = P(A)P(C)$,

$Q(B, A) = P(B, A)$,

$Q(D, C) = P(D, C)$, and

$Q(B, D) = P_s(B|\, D)P(D)$.

To obtain this distribution, the IPFP iterates 37 loops to converge with a convergence threshold 10^{-6} (the cross-entropy $I(Q_n||Q_{n-1}) < 10^{-6}$).



A	
True	**False**
0.8	0.2

A	B	
	True	**False**
True	0.9	0.1
False	0.2	0.8

E	B	
	True	**False**
True	0.35	0.65
False	0.8	0.2

C	
True	**False**
0.75	0.25

D	C	
	True	**False**
True	0.2	0.8
False	0.36	0.64

F	D	
	True	**False**
True	0.8	0.2
False	0.05	0.95

$S_D^B =$

D	B	
	True	**False**
True	0	1
False	1	0

(d) Two consistent linked BNs

A	C	B/D			
		True/True	True/False	False/True	False/False
True	**True**	0	0.5635	0.0365	0
True	**False**	0	0.1565	0.0435	0
False	**True**	0	0.0365	0.1135	0
False	**False**	0	0.0035	0.0465	0

(b) The joint distribution that identifies L_D^B and its linked BNs are consistent

Figure 4.7 An example of consistent linked BNs

Assumption 4.1: In SLBN, all linked BN are consistent.

This is the first assumption SLBN demand from those who are expected to use SLBN to model domain knowledge. Assumption 4.1 tells that the initial states of linked BNs must be reasonably compatible. Or, the linked BNs are not describing the same domain, and hence cannot be linked.

4.4 Summary

Our commonsense understanding of similarity is too vague to be formally defined and specified. In this chapter we proposed the concept *Semantic Similarity*, whose properties can be formally represented and hence utilized in intelligence systems. Variable Linkages are the concrete probabilistic representations of semantic similarities. In this chapter, first we introduced the variable linkage between pair-wise variables, which represents the simplest semantic similarities. Using pair-wise variable linkages, we discussed the semantics of variable linkages, and clarified the difference between semantic similarity and conditional dependency. A variable linkage differs from an edge in BN in that

1. A variable linkage represents semantic similarities between variables, while a BN edge represents conditional dependencies between variables.

2. The conditional probability that quantifies the semantic similarity of a variable linkage is invariant with respect to an events: $S_Y^X = P(Y \mid X) = P(Y \mid X, ev)$, where *ev* is a hard evidence in the source BN.

3. Probabilistic influences can only be propagated along the direction of variable linkages, while BN edges support probabilistic influence propagating along both directions.

We extended the definition of pair-wise variable linkage to represent similarities between two sets of variables. In this extended definition, we stated restrictions on the source and destination variables of variable linkages, and provided an informal justification for these restrictions. Formal justifications and in-depth theoretical analysis are provided in Chapter 6. Finally, we gave examples showing how to use variable linkages to represent the relationship between concepts.

In the following chapters we will provide inference methods that utilize variable linkages, reveal more features about variable linkages and give formal justifications for SLBN.

5 Evidential Inference with Variable Linkage

The Variable Linkage is defined for two purposes: to describe semantic similarity by probabilistic information, and to carry probabilistic influences from one BN to the other. In this chapter, first we present informal descriptions on how hard evidence is propagated across variable linkages. Then we propose an implementation using soft evidences and virtual evidences. The justification of this method can be found in the next chapter.

In this and the following chapters, we will propose and discuss inference methods in the scenario in which two BNs are linked by linkages of the same direction. Specifically, suppose we are given two BNs N_A, N_B along with a set of linkage L_1, …, L_n, all of which have N_A as the source BN and N_B as the destination BN. In this scenario, we can simplify the descriptions of problems, and hence clearly present our solutions. The proposed algorithm can be easily applied to scenarios involving three or more linked BNs without modifications.

5.1 Informal Descriptions of Probabilistic Influence via Linkages

In this section, we will illustrate how a hard evidence in the source BN should influence the variables in the destination BN via the variable linkages according to the semantics of variable linkages.

5.1.1 Probabilistic Influence on Destination Variables

We know, in SLBN, the prior probability distribution of the source and destination variables are consistent with its variable linkage as suggested by Assumption 4.1. So by default we have the following formula for a given linkage L_Y^X = <*X*, *Y*, N_X, N_Y, S_Y^X >:

$$P_Y(Y) = S_Y^X P_X(X), \text{ or}$$

$$P_Y(y_j) = \sum_i P_S(y_j \mid x_i) P_X(x_i).$$

If an evidence *ev* is entered into N_X, then in N_X the belief on variables *X* is updated to $P(X \mid ve)$, and the belief on variables *Y* in N_Y should be updated as

$$Q_Y(Y \mid ev) = S_Y^X P_X(X \mid ev) \quad \textbf{(5.1)}$$

This probabilistic influence is explained by Figure 5.1.

Because S_Y^X is not defined in either N_X or N_Y, we define a probability distribution $P_S(X_S, Y_S)$ for S_Y^X, where X_S and Y_S are variables identical to *X* and *Y* respectively, and initially $P_S(X_S, Y_S) = P_S(Y_S|X_S) P_X(X)$, where $P_S(X \mid Y) = P_S(Y_S|X_S)$. Because variables X_S are semantically identical to the variables *X* in N_X, after we update $P_X(X)$ to $P_X(X \mid ve)$, we use $P_X(X \mid ve)$ as an external observation of X_S and apply it as a soft evidence to X_S. Then belief on Y_S is updated by Jeffrey's rule as:

$$Q_S(Y_S) = \sum_X P_S(Y_S \mid X_S) Q_S(X_S \mid ev),$$

where $Q_S(X_S \mid ve) = P_X(X \mid ve)$. Here because S_Y^X is invariant w.r.t. events in N_X, Jeffrey's rule can always apply. For the same reason, we use $Q_S(Y_S)$ as an external observation and update Y using the soft evidence method. So finally we have

$$Q_Y(Y \mid ev) = Q_S(Y_S) = \sum_X P_S(Y_S \mid X_S) Q_S(X_S \mid ev) = S_Y^X P_X(X \mid ev).$$

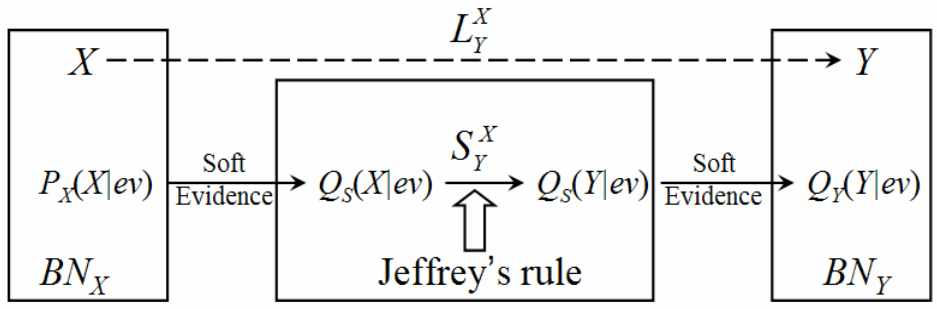


Figure 5.1 Probabilistic influence from the source variables to the destination variables via a variable linkage

Although the value of $Q_Y(Y \mid ev)$ can be calculated by equation (5.1), it cannot be directly represented in the destination BN. As we have discussed in Chapter 3, virtual evidence and soft evidence are both capable of representing external uncertain evidential findings. So here we can use a soft evidence *se* to represent the updated result of the destination variables Y as

$$P_Y(Y \mid se) = Q_Y(Y \mid ev). \quad \textbf{(5.2)}$$

5.1.2 Probabilistic Influence on Other Variables

Now we can conduct evidential update from source variables to destination variables over a variable linkage, and in this section we will discuss how the other variables should be

updated. First, we use a theorem to describe how a variable is influenced by other hard evidences in a single BN.

Theorem 5.1: In a BN, belief on variable *A* is updated if and only if

(1) one of *A*'s descendents is instantiated, or

(2) one of *A*'s ancestors is instantiated, or

(3) one of *A*'s ancestors' descendents is instantiated, or

(4) *A* is instantiated.

Proof.

Sufficiency:

Suppose in the given BN, variable *B* is instantiated to one of its state and this belief change updates the belief on *A*. Then

(1) If *B* is *A*'s descendent, then it is case a).

(2) If *B* is *A*'s ancestor, then it is case b).

(3) If *A* and *B* are not in a path but share any descendant, then one of the shared descendants must be instantiated so that *A* and *B* are d-connected. Then this is also case b).

(4) If *A* and *B* are not in a path but share any ancestor, then this is case c).

(5) Else, *B* is *A*, and this is case d).

Necessity:

From the semantics of BN, we know that in any of the 4 cases, the belief on variable *A* is

updated.

We call situation a) and d) as A is influenced by a hard evidential finding *from bottom*, and situation b) and c) as A is influenced by a hard evidential finding *from top*.

Using Theorem 5.1, we can show by equations that in what manner the probabilistic influences are propagated from the source variables to the destination variables. Let E denote the hard evidences in the source BN N_X, $L_Y^X = \langle X, Y, N_X, N_Y, S_Y^X \rangle$ denote a linkage between N_X and N_Y, and M_X denotes the variables in the Markov blanket of variables X. Then because M_X d-separate X from the other variables, the belief on X is updated by E as

$$P(X \mid E) = \sum_i P(X \mid M_X = m_i) P(M_X = m_i \mid E), \quad \textbf{(5.3)}$$

where m_i denotes a state configuration of M_X. Using equations (5.1) and (5.3) we have

$$\begin{aligned} Q(Y|E) &= S_Y^X P(X|E) \\ &= S_Y^X \sum_i P(X \mid M_X = m_i) P(M_X = m_i \mid E) \\ &= \sum_j \sum_i P_S(Y \mid X = x_j) P(X = x_j \mid M_X = m_i) P(M_X = m_i \mid E) \\ &= \sum_i P(M_X = m_i \mid E) \sum_j P_S(Y \mid X = x_j) P(X = x_j \mid M_X = m_i). \end{aligned}$$

Since $S_Y^X = P(Y \mid X)$ is invariant w.r.t. instantiations of M_X, we have

$$Q(Y \mid E) = \sum_i P(Y \mid M_X = m_i) P(M_X = m_i \mid E), \quad \textbf{(5.4)}$$

which shows that Y is influenced by the hard evidences in N_X through the Markov blanket of X. Specifically, when the hard evidences are from the top of X, then the parents of X, p_X, d-separate X from the evidences. Then equation (5.3) changes to

$$P(X \mid E) = \sum_i P(X \mid p_X = p_i) P(p_X = p_i \mid E) .$$

And similar to the derivation of equation (5.4), we have

$$Q(Y \mid E) = \sum_i P(Y \mid p_X = p_i) P(p_X = p_i \mid E) . \quad \textbf{(5.5)}$$

Similarly, when the hard evidences are from the bottom of X, then the children and the parents of the children of X, l_X, d-separate X from the evidences. Then we have

$$P(X \mid E) = \sum_i P(X \mid l_X = l_i) P(l_X = l_i \mid E) , \text{ and}$$

$$Q(Y \mid E) = \sum_i P(Y \mid l_X = l_i) P(l_X = l_i \mid E) . \quad \textbf{(5.6)}$$

As we have mentioned in Section 2.1, $M_X = p_X + l_X$. So when the hard evidences influence X from both top and bottom, all variables in the M_X are used as equation (5.4) shows.

From equations (5.3) – (5.5) we can conclude that no matter the hard evidences influence source variables from top, from bottom, or from both sides, *the probabilistic influences are propagated to the source and destination variables via the same set of variables*. Moreover, because the semantic similarity is invariant with respect to the states of the linked BNs, so, given a variable linkage, the belief change of its source variables always leads to the belief change of its destination variables, and the belief change of destination variables always follow the same quantification. Therefore, *upon a hard evidence, the source variables and destination variables are updated together and act as*

if they were representing the same concept. This "as-if" statement is still informal, a more rigorous justification will be provided in the next chapter.

Now we can tell how the evidence should be propagated in these three cases.

Case 1:

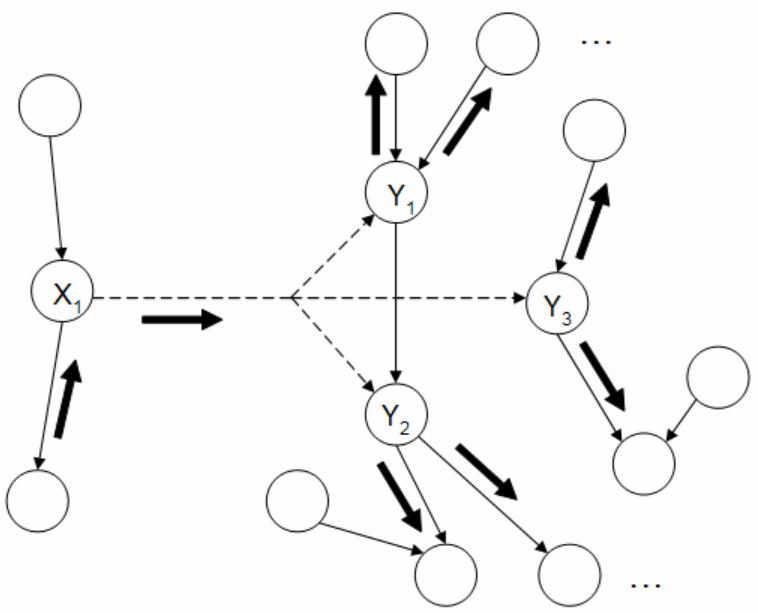


Figure 5.2 Probabilistic influence from a variable linkage: Case 1.

In Figure 5.2, black arrow stands for the propagation of the probabilistic influence from bottom. Suppose we have a linkage L_Y^X = <*X*, *Y*, N_X, N_Y, S_Y^X >. Evidence *ev* influences *X* from bottom and no evidence influences *X* from top. From equation 5.6 we know through L_Y^X , *ev* also influences *Y* from bottom. Then, in the same way as in regular BN, *ev*'s probabilistic influence are propagated as the following equations shows:

$$P(p_Y \mid ev) = \sum_Y P(p_Y \mid Y)P(Y \mid ev) \text{, and} \qquad \textbf{(5.7)}$$

$$P(l_Y \mid ev) = \sum_Y P(l_Y \mid Y)P(Y \mid ev) \text{ .} \qquad \textbf{(5.8)}$$

If we use a soft evidence *se* to update Y as suggested by equation (5.2), both p_Y and l_Y can be correctly influenced:

$$P(p_Y \mid se) = \sum_Y P(p_Y \mid Y)P(Y \mid se) \text{, and} \quad \textbf{(5.9)}$$

$$P(l_Y \mid se) = \sum_Y P(l_Y \mid Y)P(Y \mid se) \quad \textbf{(5.10)}$$

Case 2:

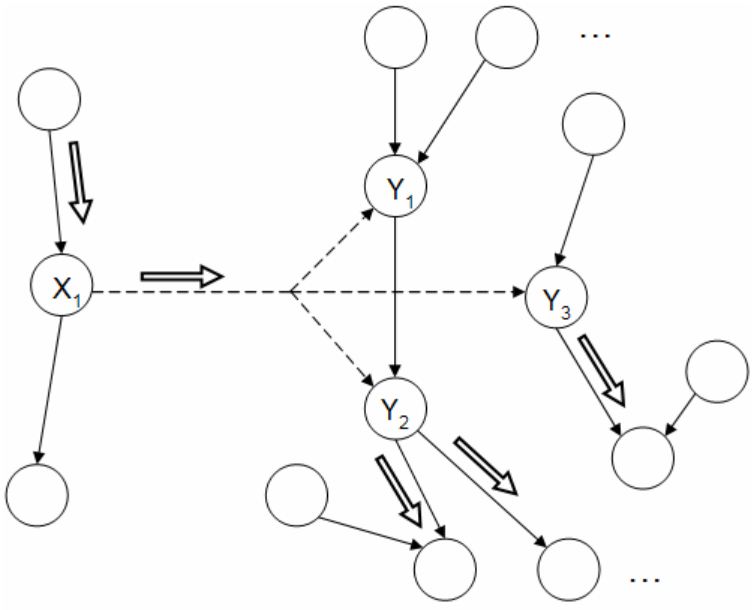


Figure 5.3 Probabilistic influence from a variable linkage: Case 2.

In Figure 5.3, white arrows stand for the propagation of the probabilistic influence from top. Suppose we have a linkage L_Y^X = <X, Y, N_X, N_Y, S_Y^X >. Evidence *ev* influences X from top and no evidence influences X from bottom. From equation (5.5) we know through L_Y^X , *ev* also influences Y from top. Then *ev*'s probabilistic influence are propagated as the following equations shows:

$$P(p_Y \mid ev) = P(p_Y) \text{, and} \quad \textbf{(5.11)}$$

$$P(l_Y \mid ev) = \sum_Y P(l_Y \mid Y) P(Y \mid ev) . \quad \textbf{(5.12)}$$

If we use soft evidence *se* to update Y as suggested by equation (5.2), l_Y can be correctly influenced, but p_Y would also be updated by soft evidence using equation (5.9). So in the inference process, we need to apply additional soft evidence to variables in p_Y so that the influence of *se* is eliminated. That is to find se_p for the variables p_Y such that

$$P(p_Y \mid se, se_p) = P(p_Y) . \quad \textbf{(5.13)}$$

Case 3:

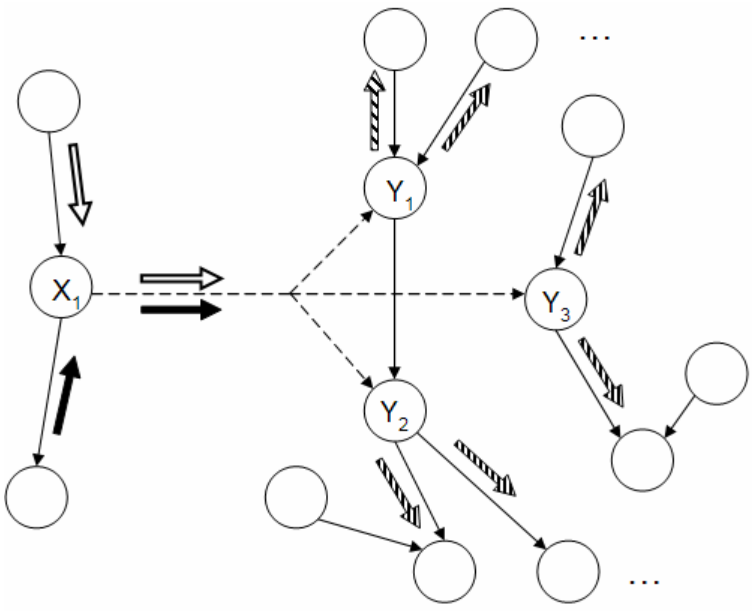


Figure 5.4 Probabilistic influence from a variable linkage: Case 3.

In Figure 5.4, black arrows stand for the propagation of the probabilistic influence from bottom, white arrows stand for the propagation of the probabilistic influence from top, and striped arrows stand for the propagation of the combined probabilistic influences. Suppose we have a linkage L_Y^X = <X, Y, N_X, N_Y, S_Y^X >. Evidence ev_1 influences X from

top and evidence ev_2 influence X from bottom. Then through L_Y^X , ev_1 influences Y from top and ev_2 influences Y from bottom, ev_1 and ev_2 may or may not be the same evidence. In this case, ev_1 and ev_2's probabilistic influences are propagated as the following equations shows:

$$P(p_Y \mid ev_1, ev_2) = \sum_Y P(p_Y \mid Y, p_X) P(Y, p_X \mid ev_1, ev_2) \text{ , and} \quad \textbf{(5.14)}$$

$$P(l_Y \mid ev_1, ev_2) = \sum_Y P(l_Y \mid Y) P(Y \mid ev_1, ev_2) . \quad \textbf{(5.15)}$$

Similar to case 2, if we use soft evidence se to update Y, l_Y can be correctly influenced, but p_Y would be updated by soft evidence using equation (5.9). So in the inference process, we need to apply additional soft evidence to p_Y to correct the influence of se. So we want to find se_p for the variables p_Y such that

$$P(p_Y \mid se, se_p) = P(p_Y \mid ev_1, ev_2) . \quad \textbf{(5.16)}$$

5.2 Evidential Update on Destination BN

After deriving equations (5.4) – (5.6) for passing probabilistic influence from the source BN to the destination BN through variable linkages, we now develop an algorithm for belief update on the destination BN. This algorithm incorporates soft evidences and IPFP to manipulate the probabilistic influence passing through variable linkages, and it can be realized as an extension of any current BN inference method. In our belief update method, inference is divided into two parts: inference inside a single BN and inference across variable linkages. The inference inside a single BN is the same as what is done by regular BN inference method. The method of inference across variable linkages extends the

regular BN inference methods with rules to control the probabilistic influence along the variable linkages.

5.2.1 Single Variable Linkage

First we provide an inference algorithm dealing with a single variable linkage between two BNs. This algorithm is a straightforward implementation of what was described in Section 5.1. In this algorithm, first we create a joint probability distribution for the given linkage L: $Q(p_{src}, p_{dest}, dest(L))$, where $p_{src} = parent(src(L))$, and $p_{dest} = parent(dest\ (L))$. From Assumption 4.1 we know linked BNs need to be consistent, and hence the distribution $Q(p_{src}, src(L), p_{dest}, dest(L))$ can be obtained by IPFP (see Section 4.3). Then,

$$Q(p_{src}, p_{dest}, dest(L)) = ?_{\ src(L)}\ Q(p_{src}, src(L), p_{dest}, dest(L)).$$

Similar to the process by which $Q(p_{src}, src(L), p_{dest}, dest(L))$ is obtained, $Q(p_{src}, p_{dest}, dest(L))$ can also be obtained by IPFP directly.

Next we calculate $Q(p_{src}, p_{dest}, dest(L) \mid ev)$ by entering ev's probabilistic influence to Q. The probabilistic influence from top is entered at p_{src}, and the probabilistic influences from bottom is entered at $dest(L)$. Then Q is updated by IPFP, and the updated results are entered into the destination BN.

Algorithm 5.1

Input: two linked BNs, a variable linkage, and hard evidences applied to the source BN.

Output: belief update result of the destination BN.

Methods:

function Propagate_Evidence(N_1, HE, N_2, L)

1 $Q_{Linkage}$ = **Construct_Linkage_JPD**(N_1, N_2, L);

2 **Update_Linkage_JPD** (N_1, HE, L, $Q_{Linkage}$);

3 **Update_Destination_BN**(L, $Q_{Linkage}$, N_2);

function Construct_Linkage_JPD (N_1, N_2, L) **return** a Joint Probability Distribution

1 $p_{src} = parent(src(L))$; $p_{dest} = parent(dest(L))$;

2 $Q_0(p_{src},p_{dest}, dest(L)) = \{d(p_{src}, p_{dest}, dest(L))^{-1}\}$; //initiate to uniform dist.

3 $Q(p_{src},p_{dest}) = P_1(p_{src})\ P_2(p_{dest})$; // assume p_{src} and p_{dest} are independent

4 $Q(p_{src}, dest(L)) = ?\ _{src(L)} P_S(dest(L)|src(L))P_1(src(L), p_{src})$; //equation (5.2)

5 $Q(p_{dest}, dest(L)) = P_2(p_{dest}, dest(L))$;

6 $Q_{Linkage}(p_{src},p_{dest}, dest(L)) =$

IPFP on Q_0 $(p_{src},p_{dest}, dest(L))$

with constraints $Q(p_{src},p_{dest})$, $Q(p_{src}, dest(L))$, and $Q(p_{dest}, dest(L))$;

7 **return** $Q_{Linkage}(p_{src},p_{dest}, dest(L))$;

function Update_Linkage_JPD (N_1, HE, L, $Q_{Linkage}$)

1 *influenceType* = Check_Influence_Type(N_1, L);

2 $P(p_{src} \mid HE) = P_1(parent(src(L)) \mid HE)$; // by regular BN inference

3 $P(dest(L) \mid HE) = ?\ _{src(L)} P_S(dest(L)|src(L))\ P_1(src(L) \mid HE)$; //Jeffrey's rule

4 if (*influenceType* == FROM_TOP)

5 $$Q_{Linkage}(p_{src}, p_{dest}, dest(L)) = Q_{Linkage}(p_{src}, p_{dest}, dest(L)) \frac{P(p_{src} \mid HE)}{Q_{Linkage}(p_{src})};$$

6 if (*influenceType* == FROM_BOTTOM)

7 $$Q_{Linkage}(p_{src}, p_{dest}, dest(L)) = Q_{Linkage}(p_{src}, p_{dest}, dest(L)) \frac{P(dest(L) \mid HE)}{Q_{Linkage}(dest(L))};$$

8 if (*influenceType* == FROM_BOTH)

9 $Q_0(p_{src}, p_{dest}, dest(L)) = Q_{Linkage}(p_{src}, p_{dest}, dest(L))$;

10 $Q_{Linkage}(p_{src}, p_{dest}, dest(L))$ =

IPFP on $Q_0(p_{src}, p_{dest}, dest(L))$

with constraints $P(dest(L) \mid HE)$ and $P(p_{src} \mid HE)$);

function Update_Destination_BN(L, $Q_{Linkage}$, N_2)

1 Set_Soft_Evidence(N_2, $dest(L)$, $Q_{Linkage}$ ($dest(L)$));

2 if (L.influenceType== FROM_TOP || FROM_BOTH)

3 $p_{dest} = parent(dest(L))$;

4 Set_Soft_Evidence (N_2, p_{dest}, $Q_{Linkage}$ (p_{dest}));

5 Soft_Evidential_Update(N_2);

The function Set_Soft_Evidence(N, X, Q) set a soft evidence of target distribution Q on variables X in BN N. The entered soft evidence does not cause belief update until the function Soft_Evidential_Update(N) is called.

$Q_{Linkage}$ =

A	C	D	
		True	False
True	True	0.0365	0.5635
True	False	0.0435	0.1565
False	True	0.1135	0.0365
False	False	0.0465	0.0035

Figure 5.5 The $Q_{Linkage}$ for linkage L_D^B in Figure 4.9

Now we explain our algorithm using the example in Figure 4.9. Figure 5.5 depicts the $Q_{Linkage} = Q(A, C, D)$ for the linkage L_D^B. The full joint probability distribution $Q(A, C, B, D)$ can be found in Figure 4.9. Here we present three test cases:

1) If we instantiate *A* to state *True*, then in function Update_Linkage_BN() the probabilistic influence is entered to $Q_{Linkage}$ as $Q(A) = (1.0, 0)$, and in function Update_Destination_BN() the soft evidence applied to the destination BN is $Q(C) = (0.75, 0.25)$ and $Q(D) = (0.1, 0.9)$.

2) If we instantiate *E* to the state *True*, then in function Update_Linkage_BN() the probabilistic influence is entered to $Q_{Linkage}$ as $Q(D) = (0.419, 0.581)$, and in function Update_Destination_BN() the soft evidence applied to the destination BN is $Q(D) = (0.419, 0.581)$.

3) If we instantiate *A* to the state *True* and *E* to the state *True*, then in function Update_Linkage_BN() the probabilistic influence is entered to $Q_{Linkage}$ as $Q(A) = (1.0, 0)$ and $Q(D) = (0.202, 0.798)$, and in function Update_Destination_BN() the soft evidence applied to the destination BN is $Q(C) = (0.717, 0.283)$ and $Q(D) = (0.202, 0.798)$.

5.2.2 Multiple Variable Linkages

When multiple linkages are present, belief update in the destination BN becomes more complicated because some variables can be affected by probabilistic influences from more than one linkages. For example, suppose we have edges A? B and A? C in the destination BN, $B \in dest(L_1)$, and $C \in dest(L_2)$, when an evidential influence is propagated using both L_1 and L_2, the belief on A cannot be updated by Update_Destination_BN() using either one of the two linkage, but should be updated by the accumulated influences from both L_1 and L_2. Therefore, additional rules should be applied in Update_Destination_BN() to accumulate the influence from different linkages.

We know soft evidence only describes the result of an external probabilistic influence, so it is proper to use it to update the belief on the destination variables $dest(L)$ because the belief on the destination variables are determined only by external probabilistic influences from the source BN. And to accumulate probabilistic influence from different sources, we use likelihood ratios to represent how the external probabilistic influences update the target variables rather than use soft evidences to specify the belief update results. The update result of likelihood ratios can be accumulated using the Bayes inference of BN. As we had stated in Chapter 3, soft evidence and virtual evidence are equivalent in expressing external observations. In the Case 2 discussed in Section 5.1.2, when we use virtual evidences to represent soft evidences, to satisfy equation (5.11) and (5.12), not considering the influences from other linkages, we need to find and apply ve to Y and ve_p to p_Y such that

$$P(p_Y \mid ve, ve_p) = P(p_Y \mid ev) = P(p_Y), \text{ and} \qquad \textbf{(5.17)}$$

$$P(l_Y \mid ve, ve_p) = P(l_Y \mid ev) = \sum_Y P(l_Y \mid Y)P(Y \mid ev)\,. \quad \textbf{(5.18)}$$

To achieve (5.17) and (5.18), an iteration method is used to find *ve* and ve_p:

$$lh(ve_0) = P(Y \mid ev) \cdot P(Y)^{-1}\,, \quad \textbf{(5.19)}$$

Iterate the following equations until $lh(ve_j)$ converges:

$$lh(ve_j) = P(Y \mid ev) \cdot P(Y \mid ve_{j-1}, ve_p)^{-1} \cdot lh(ve_{j-1})\,, \text{ and} \quad \textbf{(5.20)}$$

$$lh(ve_p) = (\textstyle\sum_Y P(Y \mid p_Y) lh(ve_j))^{-1}\,, \quad \textbf{(5.21)}$$

where $lh(ve)$ is the likelihood ratio of virtual evidence *ve*. Equation (5.21) is derived from (5.17) by the following steps:

$$P(p_Y \mid ve, ve_p) = P(p_Y)$$

$$\Leftrightarrow P(p_Y \mid ve) \cdot lh(ve_p) = P(p_Y) \qquad \text{(from equation (3.5))}$$

$$\Leftrightarrow \textstyle\sum_Y P(p_Y \mid Y)P(Y \mid ve) \cdot lh(ve_p) = P(p_Y)$$

$$\Leftrightarrow \textstyle\sum_Y \frac{P(Y \mid p_Y)P(p_Y)}{P(Y)} P(Y \mid ve) \cdot lh(ve_p) = P(p_Y)$$

$$\Leftrightarrow \textstyle\sum_Y \frac{P(Y \mid p_Y)}{P(Y)} P(Y \mid ve) \cdot lh(ve_p) = 1$$

$$\Leftrightarrow \textstyle\sum_Y P(Y \mid p_Y) lh(ve) \cdot lh(ve_p) = 1 \qquad \text{(from the definition of likelihood ratio)}$$

$$\Leftrightarrow lh(ve_p) = (\textstyle\sum_Y P(Y \mid p_Y) lh(ve))^{-1}$$

In each iteration step, virtual evidence ve_j tries to modify the distribution of *Y* to $P(Y|ev)$, while the virtual evidence ve_p can eliminate the probabilistic influence from *Y* to

p_Y but shifts Y off its target distribution at the same time. So we need to iterate, and if this iteration converges, it stops when equation (5.18) is satisfied.

The iteration of equation (5.19) – (5.21) solves the probabilistic influence problem in our observation case 2, and a similar method could be applied for case 3. In the Case 3 of Section 5.1.2, we need to find *ve* and ve_p which satisfy equations (5.14) and (5.15), such that

$$P(p_Y \mid ve, ve_p) = P(p_Y \mid ev_1, ev_2) = \sum_Y P(p_Y \mid Y, p_X) P(Y, p_X \mid ev_1, ev_2), \quad \textbf{(5.22)}$$

$$P(l_Y \mid ve, ve_p) = P(l_Y \mid ev_1, ev_2) = \sum_Y P(l_Y \mid Y) P(Y \mid ev_1, ev_2). \quad \textbf{(5.23)}$$

Equation (5.23) and (5.18) are fairly the same. The difference between (5.22) and (5.17) is that in (5.22), ve_p eliminates the influence from Y to p_Y, while in(5.17), ve_p not only eliminates the influence from Y to p_Y, but also adds the correct influence from Y to p_Y. To achieve (5.22) and (5.23), an iteration method is used to find *ve* and ve_p:

$$lh(ve_0) = P(Y \mid ev_1, ev_2) \cdot P(Y)^{-1}, \quad \textbf{(5.24)}$$

Iterate the following equations until $lh(ve_j)$ converges:

$$lh(ve_j) = P(Y \mid ev_1, ev_2) \cdot P(Y \mid ve_{j-1}, ve_p)^{-1} \cdot lh(ve_{j-1}), \text{ and} \quad \textbf{(5.25)}$$

$$lh(ve_p) = (\textstyle\sum_Y P(Y \mid p_Y) lh(ve_j))^{-1} \cdot P(p_Y \mid ev_1, ev_2) \cdot P(p_Y)^{-1}, \quad \textbf{(5.26)}$$

Algorithm 5.2

Input: Two linked BNs, a set of variable linkages, and hard evidences applied in the source BN.

Output: Propagate the hard evidences from source BN to the destination, and returns the belief update result of the destination BN.

Methods:

function Propagate_Evidence(N_1, *HE*, N_2, L[])

1 for i=1 to sizeof(L[])

2 $Q_{Linkage}[i]$ = **Construct_Linkage_JPD**(N_1, N_2, $L[i]$);

3 **Enter_Evidence**(N_1, *HE*);

4 for i=1 to sizeof(L[])

5 **Update_Linkage_JPD** (N_1, $L[i]$, $Q_{Linkage}[i]$);

6 **Update_Destination_BN**(L[], $Q_{Linkage}$[], N_2);

function Update_Destination_BN(L[], $Q_{Linkage}$[], N_2)

1 d = 1; i = 0; j = 0;

2 p_{dest} = *parent*(*dest*($L[i]$)); X is the set of variables in N_2;

3 Q_0 (X)= $P_2(X)$;

4 while (d > *threshold*)

5 $lh_{dest} = Q_{Linkage}(dest(L[i])) \cdot Q_j(dest(L[i]))^{-1}$;

6 Set_Likelihood_Ratio(N_2, *dest*($L[i]$), lh_{dest});

7 **if** ($L[i]$.influenceType == FROM_TOP || FROM_BOTH)

8 $lh_p = (?_{\ dest(L[i])} Q_j(dest(L[i])|\ p_{dest})\ lh_{dest})^{-1}$; //equation (5.21)

9 if ($L[i]$.influenceType== FROM_BOTH)

10 $lh_p = lh_p \cdot Q_j(p_{dest}) \cdot P_2(p_{dest})^{-1}$; //equation (5.26)

11 Set_Likelihood_Ratio(N_2, p_{dest}, lh_p)

12 $Q_{j+1}(X)$ = Virtual_Evidential_Update(N_2);

13 d = distance($Q_j(X)$, $Q_{j+1}(X)$);

14 $j = j+1$; $i = j$ **mod** sizeof($L[]$);

5.3 Summary

In this chapter, we depict how the hard evidences in the source BN are propagated to influence the variables in the destination BN via variable linkages, and then give implementations utilizing soft evidences and virtual evidences.

By equation (5.1) we state that the belief on the destination variables is determined by the belief on the source variables, and by equations (5.4) – (5.6) we tell that the evidential probabilistic influences are propagated to the destination variables through the same set of variables as to the source variables. By differentiating how the probabilistic influences are entered to the source variables, we know how the destination BN should be updated. When the hard evidences influence the source variables from top, the destination BN is updated using equations (5.7) and (5.8). When the hard evidences influence the source variables from bottom, the destination BN is updated using equations (5.11) and (5.12). When the hard evidences influence the source variables from both sides, the destination BN is updated using equations (5.14) and (5.15).

The probabilistic influences from the source BN to the destination BN are viewed as soft evidences in the destination BN which are further represented as virtual evidences in our algorithms. We calculate the results of these probabilistic influences in a joint probability distribution $Q_{Linkage}$, which includes the variable linkage, the source and destination variables and their parents. Then the belief update is firstly calculated over

$Q_{Linkage}$, the results are then applied to the destination BN. When multiple variable linkages are used together, soft evidences are represented by virtual evidences, which are carefully manipulated to accumulate the probabilistic influences from different linkages.

The inference algorithms proposed in this chapter is formally justified in Chapter 6 with the help of J-Graph.

6 J-Graph for SLBN

The semantic similarity between variables in different BNs is a type of relation beyond the semantics of the BN edge, so how to justify the proposed inference methods which involves two types of relations is an important issue.

In Section 6.1 we will show that the global knowledge by which the SLBN is based on is generally not accessible, but we still can join the knowledge of each linked BNs together using variable linkages. However, this joint knowledge is not in the form of BN. In Section 6.2 we will make some assumptions which can be used to justify the inference methods we proposed in Chapter 5. These assumptions can further support us in constructing the joint knowledge for a SLBN. This joint knowledge of SLBN, named J-graph, is proposed in Section 6.3 and its construction method is given in Section 6.4. In Section 6.5, we justify that J-graph correctly encodes the variable linkages by showing that the inference on J-graph is exactly the same as the inference on SLBN.

6.1 The Unaccessible Global Knowledge of SLBN

Now we consider answering the question that under what circumstances variable linkages can be created between given BNs. In MSBN, all linked networks are required to form a unique global Bayesian network, which is a very restrictive requirement and is not achievable in many use cases. In contrast, in SLBN, linked BNs are assumed to be separately developed rather than sectioned from a single BN. Besides, neither does SLBN assume that communication can always be carried between any BNs because of the

directionality of the variable linkages. Moreover, it is clear that inference cannot be correctly conducted if these independently constructed BNs model the causalities in the domain substantially differently. Therefore, we want some formal measurements or criteria to check if the given BNs and linkages can utilize SLBN's inference methods. One criterion could be to check if a global knowledge can be constructed comprising the separately developed BNs' probability distributions and conditional interdependencies of the given variable linkages. This global knowledge must conform with the semantics of SLBN and support the inference methods of SLBN.

The ideal solution is that the global knowledge could be in form of a BN and each linked BN is the projection of the global BN on an aspect of domain knowledge. More precisely, it would be ideal if there exists a global Bayesian network for the linked BNs such that:

1 every variable in the linked BN has a representation in the global BN;

2 the probability distribution of each linked BN is the marginalization of the probability distribution of the global BN; and

3 the conditional probability in each variable linkage is also the marginalization of the probability distribution of the global BN.

However, even if this global BN exists, it may not be uniquely identified by the knowledge encoded in BNs and variable linkages. In general, the global BN for a given SLBN is not accessible because the linked BNs do not provide enough information to construct a global BN for the following reasons:

1 a variable linkage quantifies how the source variables is similar to the destination variables, but the similarity of the reverse direction may not be available;

2 it is unknown that how to joint the linked variables as they have separate CPTs and different local parents;

3 the variables that are located in different local BNs and are not similar may have casual influences to each other, and since they are in different local BNs, their interdependencies are not directly captured by the SLBN.

So, we can see that the SLBN are all we know about this domain and the inference of SLBN is reasoning on this available domain knowledge.[5]

Although the global knowledge is not accessible, it does exist for the given SLBN, or the inference in SLBN would be impossible. In the scenario of SLBN, each of the linked BNs encodes only an aspect of the global knowledge and all variables and the dependencies between the variables in the linked BNs are theoretically derived from a global knowledge. And because of this, the linked BNs and the variable linkages need to be consistent to some extent, and such consistency requires the linked BNs and the linkages must obey some restrictions. The Assumption 4.1 is one of these restrictions. In the next section, more restrictions are expressed as assumptions on SLBN.

6.2 Assumptions for SLBN

In this section we claim some restriction about the linked BN and variable linkages to ensure the lined BNs and the variable linkages can be joined into a global knowledge.

[5] However, from another point of view, if the global BN is already known, many other approaches could be applied, such as MSBN, and such problem is beyond the discussion in this paper.

These restrictions are claimed in the form of assumptions and will not be justified. But they are reasonable in the semantics of BN.

Assumption 6.1: In SLBN, one variable can be the source variable or destination variable of only one variable linkage between a given pair of BNs.

Variable linkages cannot share source variables and destination variables. If variable *A* is similar to variable *B* and variable *C*, then *A* is also similar to the union of *B* and *C*. So rather than creating two linkages from *A* to *B* and *A* to *C* (or from *B* to *A* and *C* to *A*), SLBN require a single linkage from *A* to both *B* and *C* (or from *B* and *C* to *A*).

If Assumption 6.1 is violated, the linked BNs may also be able to join to a global knowledge. However, Assumption 6.1 will greatly reduce the complexity of the joining process. In other words, Assumption 6.1 is an engineering assumption. In Section 6.4.1 we will see that with the support of Assumption 6.1, each variable in the SLBN will only have one representation in the jointed knowledge.

Assumption 6.2: In SLBN, all linked BNs have the same causality. Specifically, given two linked BNs N_X and N_Y, any path *path*(*A*, *B*) in N_X and *path*(*C*, *D*) in N_Y can interpret the domain at the same time.

The *causality of a BN* refers to the graph structure by which the BN interpret the relationship between the variables of the domain. According to the semantics of Bayesian network, one distribution can be encoded by multiple different BNs, all of which represent same conditional interdependencies using different structures. Assumption 6.2 requires only BNs that have identical interpretation about domain knowledge can be jointed together. For example, give a probability distribution *P*(*Diabetes*, *Hyperglycemia*),

we can use two graph structures to build a BN to encode this distribution: *Diabetes* ? *Hyperglycemia*, and *Hyperglycemia* ? *Diabetes*. These different graph structures present different interpretations for the domain. The graph structure *Diabetes* ? *Hyperglycemia* presents the causal relation in which diabetes is the cause of hyperglycemia, while the graph structure *Hyperglycemia* ? *Diabetes* presents the abductive relation in which hyperglycemia causes the suspicion of diabetes. Both of these graph structures are good for their own purpose, but cannot co-exist in the same BN.

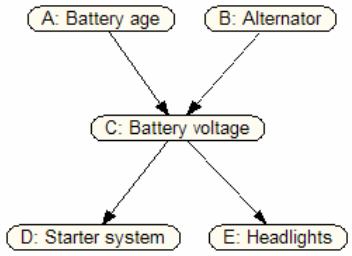
Since all linked BNs are derived from a global knowledge and describe different aspects of the same domain, all paths from them should originate from the same model, and hence should interpret the domain at the same time.

Assumption 6.2 is not really operational and cannot be checked within SLBN. However, it can be checked by a domain expert.

Assumption 6.3 (Interdependencies between variables across variable linkages): Given linkage $L_Y^X = <X, Y, N_X, N_Y, S_Y^X>$, and suppose N_X encodes casual sequences by *path*(A, X_1) and *path*(X_1, B), where $X_1 \in X$, A and B are variables in N_X. Then $\forall Y_i \in Y$, create *path*(A, Y_i) and *path*(Y_i, B), and they can interpret the domain with all paths in N_Y at the same time.

Assumption 6.3 actually claims that linked variables have the same causes and effects as if they were representing the same concept. This is consistent with the claim that the linkages are invariant with respect with the other evidences. As has been discussed in Section 5.1.2, the belief on the destination variables are always updated at the same time as the source variables and are always chaged following the same quantification.

Figure 6.1 depicts an example explaining Assumption 6.3. In Figure 6.1 (a) we present a BN modeling the diagnosis of a car's battery subsystem. The SLBN derived from this BN is depicted in Figure 6.1 (a) by SLBN. Figure 6.1 (b) depicts the derived SLBN, where a linkage is created for identical variables *C* and *C'*. Note that the given linkage's source and destination, *C* and *C'* in Figure 6.1 (b), are represented by node *C* in the original BN of Figure 6.1 (a). Figure 6.1 (c) depicts another two linked BNs which try to model the same domain. However, in Figure 6.1 (c) the destination BN has a different interpretation about the domain than the source BN. Clearly, *path*(*A*, *C*) in the source BN and *path*(*E*, *C*) in the destination cannot be consistent interpretations of the domain as they represent contrary causalities. So we say the linked BNs in Figure 6.1(c) are not a valid SLBN according to Assumption 6.3.



Battery Age		
new	**old**	**very old**
0.4	0.4	0.2

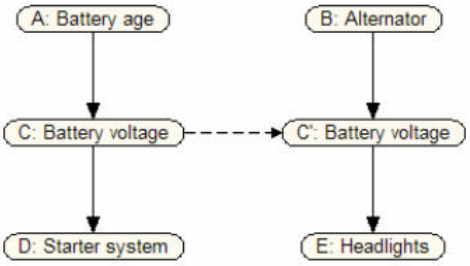
Alternator	
Okay	**Faulty**
0.997	0.003

BatVolt	**light**		
	bright	**dim**	**off**
strong	0.94	0.01	0.05
weak	0	0.95	0.5
dead	0	0	1

Alter	**BatAge**	**Battery voltage**		
		strong	**weak**	**dead**
Okay	**new**	0.479	0.17	0.351
Okay	**old**	0.402	0.175	0.423
Okay	**very old**	0.301	0.2	0.499
Faulty	**new**	0.008	0.30	0.692
Faulty	**old**	0.004	0.2	0.796
Faulty	**very old**	0.002	01	0.898

BatVolt	**Starter**	
	Okay	**Faulty**
strong	0.98	0.02
weak	0.90	0.10
dead	0.10	0.90

(a) A car diagnosis BN.

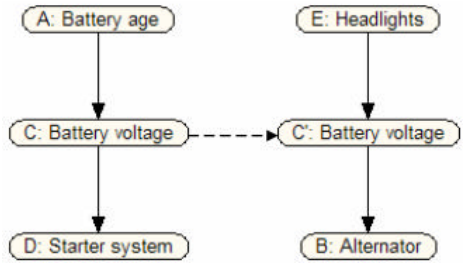


BatAge	C: Battery voltage		
	strong	weak	dead
new	0.478	0.170	0.352
old	0.401	0.175	0.424
very old	0.300	0.200	0.500

Alter	C': Battery voltage		
	strong	weak	dead
Okay	0.413	0.178	0.409
Faulty	0.005	0.220	0.775

(b) Linked BNs that are consistent with the car diagnosis BN in (a).

Unlisted CPTs are the same as what are listed in (a).



HeadLi	C': Battery voltage		
	strong	weak	dead
bright	1	0	0
dim	0.024	0.976	0
off	0.047	0.020	0.933

BatVolt	Alternator	
	Okay	Faulty
strong	1.0	0.0
weak	0.996	0.004
dead	0.994	0.006

(c) Linked BNs that are inconsistent with the car diagnosis BN in (a).

Unlisted CPTs are the same as what are listed in (a) and (b).

Figure 6.1 A car diagnosis BN and its derived Linked BNs

Theorem 6.1: In SLBN, two linkages cannot be crossed. Specifically, for two linkages L_Y^X = <X, Y, N_X, N_Y, S_Y^X > and $L_{Y'}^{X'}$ = <X', Y', N_X, N_Y, $S_{Y'}^{X'}$ >, if there exists $X_1 \in X$,

$X_1' \in X'$ such that there exist *path*(X_1, X_1'), then all variables in X must not be the descendant of X', and all variables in Y must not be descendant of Y'.

Proof.

We prove this theorem by contradiction.

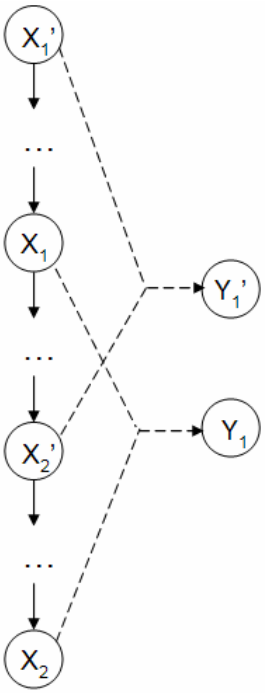
1) As shown in Figure 6.2(a), suppose there exists X_1, $X_2 \in X$, and X_1', $X_2' \in X'$, and *path*(X_1', X_1, X_2', X_2). From the definition of variable linkage (Definition 4.2) we know as long as X_1' and X_2' are in a path, then all variables between them should also be included as the source variables of L'. Therefore, this is a contradiction to the definition of variable linkage.

2) As shown in Figure 6.2(b), suppose there exists X_1, $X_2 \in X$, and X_1', $X_2' \in X'$, *path*(X_1', X_1), *path*(X_2, X_2'), and X_1', X_1 are neither ancestors nor descendants of X_2, X_2'. From Assumption 6.3 we know, because of L_Y^X , the graph structure of N_Y can interpret the domain along with *path*(X_1', Y_1, X_2'), and further with *path*(Y_1, Y_1'). Similarly, because of $L_{Y'}^{X'}$, the graph structure of N_Y can interpret the domain along with *path*(X_2, Y_1', X_1), and further with *path*(Y_1', Y_1). This is a contradiction as a BN cannot admit cyclic causal sequences.

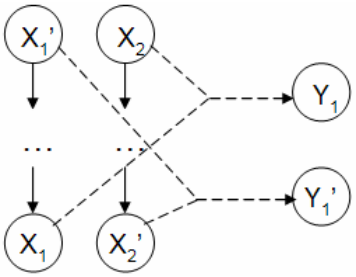
3) As shown in Figure 6.2(c), suppose there exists $X_1 \in X$, $X_1' \in X'$, and *path*(X_1, X_1'), but one variable $Y_1 \in Y$ is the descendant of one variables $Y_1' \in Y'$ as *path*(Y_1', Y_1). From Assumption 6.3 we know, because of L_Y^X , N_Y can interpret the domain along with *path*(Y_1, X_1'), and further with *path*(Y_1, Y_1'). This is a contradiction as a BN cannot admit *path*(Y_1, Y_1') and *path*(Y_1', Y_1) at the same time.

From the above reduction we can conclude that Theorem 4.2 is correct. ¦

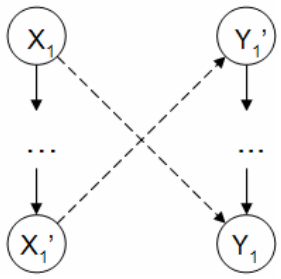
Theorem 6.1 claims that linkages cannot cross each other or the interdependency between variables across the linkages cannot be modeled. In Section 6.4, we will show Assumption 3 is the precondition to construct the joint knowledge for given SLBN.



(a) Crossed linkages of Case 1



(b) Crossed linkages of Case 2



(c) Crossed linkages of Case 3

Figure 6.2 Crossed linkages

6.3 Definition of J-Graph

While the global knowledge of a SLBN is not accessible, Joint-graph (J-graph) is proposed to capture all the domain knowledge expressed in a SLBN. A J-graph is a DAG which encodes the probabilistic dependencies between the variables in SLBN. It is converted from the linked BNs and models how the state change of variables in one BN to influence variables in another BN across the linkages. In the remaining sections of this chapter, *Q* denotes the probability distributions in the J-graph, and *P* denotes the the probability distributions in BNs.

Definition 6.1 (J-graph): A Joint-graph (J-graph) for linked BNs is a graph for which the following holds:

1. It is a directed acyclic graph (DAG).
2. It has a set of nodes of two types: *variable nodes* and *linkage nodes*. A variable node represents a variable from the linked BNs and is labeled with its original name and its original BN's name. A linkage node represents a variable linkage and its source and destination variables.

3. It contains a set of edges between nodes representing dependency relations between variables like BN's edges. For any edge *A*? *B* in the local BN, if *A* and *B* are not involved in the same linkage, the J-graph has an edge *A'*? *B'*, where *A'* and *B'* are the nodes that J-graph uses to represent *A* and *B* respectively.

4. Each node has a conditional probability table quantifies the effects its parents have on the node such that

   a) For each variable *A* in the local BN, $Q(A'|p'(A')) = P(A|p(A))$, where $p(A)$ is the parents of *A* in its original BN, where *A'* and $p'(A')$ are the nodes that J-graph uses to represent *A* and $p(A)$ respectively;

   b) The CPT on linkage node satisfies $Q(dest(L)|\, src(L)) = P(dest(L)|\, src(L))$.

Please note that, different from the node in BN, ***a node in J-graph may represent one variable (variable node) or multiple variables (linkage node)***.

6.4 Construction of J-Graph

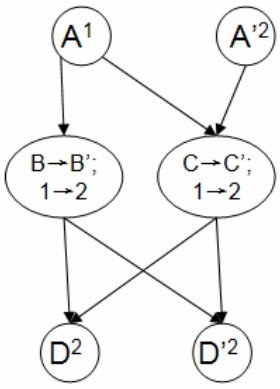
Now we give an algorithm to construct the J-graph from a given SLBN:

6.4.1 Constructing the Structure of J-graph

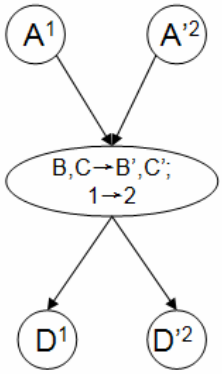
Algorithm 6.1 (Construct the structure of a J-graph): Suppose we have linked Bayesian networks $N_1, N_2, \ldots, N_n$, where $N_i = <V_i, E_i, P_i>$, V_i is the set of nodes, E_i is the set of edges, and P_i is the set of CPTs in N_i. Let V_i^* denote the nodes that are not involved with any linkages in N_i, and E_i^* denote the edges whose both ends are nodes in V_i^*. $V = \bigcup V_i$,

$V^* = \bigcup V_i^*$, $E = \bigcup E_i$, and $E^* = \bigcup E_i^*$. Then the J-graph for the linked BNs is constructed by the following steps:

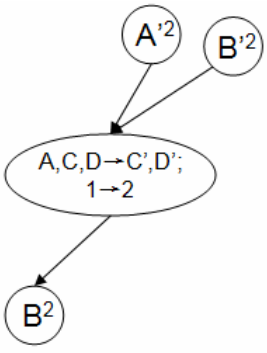
1. Initialize J-graph for the linked BNs as empty;
2. For each node $v^* \in V^*$, create a variable node in the J-graph, and label the variable with v^*'s original name with the index of its origin BN as its label's superscript;
3. For each edge $e^* \in E^*$, create an edge between the corresponding variables in the J-graph.
4. For each linkage, create a linkage node for it and label it as: "*source nodes* ? *destination nodes*; *source BN* ? *destination BN*", e.g. the linkage node for <{X_1, X_2},{Y_1}, N_1, N_2, S> should be "X_1, X_2 ? Y_1; 1 ? 2".
5. For each edge $e \in E - E^*$,
   a) if e is from a node $v^* \in V^*$ to a node $v \in V - V^*$, and a linkage L involves v in either *src*(L) or *dest*(L), create an edge from the variable node for v^* to the linkage node for L, if there does not exist one; or
   b) if e is from a node $v \in V - V^*$ to a node $v^* \in V$, and a linkage L involves v, create an edge from the linkage node for L to variable node for v^*, if there does not exist one; or
   c) if e is from a node $v_1 \in V - V^*$ to a node $v_2 \in V - V^*$, a linkage L_1 involves v_1 and a linkage L_2 involves v_2, create an edge from the linkage node for L_1 to the linkage node for L_2, if there does not exist one.



(a) J-graph for Figure 4.1(a)



(e) J-graph for Figure 4.1(b)



(c) J-graph for Figure 4.1(d)

Figure 6.3 J-graphs for linked BNs in Figure 4.1

Figure 6.3 presents the J-graphs for the linked BNs in Figure 1.

For convenience, we propose the terms principled ancestors and principled descendants for variables in local BNs. For every variable *v* in every local BN, the variables that are not known or not encoded as the ancestors of *v* by the local BN but are encoded as the ancestors of *v* in the J-graph is called the *principled ancestors* of *v*, and the variables that are not known or not encoded as the descendant of *v* by the local BN but are encoded as the descendant of *v* in J-graph is called the *principled descendants* of *v*.

Lemma 6.1: For ever edge *e*: *A*? *B* in every linked BN, if *A* and *B* are not included in the same linkage, the J-graph generated by Algorithm 6.1 has one edge *e'*: *A'*? *B'* such that *A'* and *B'* are J-graph nodes that represent *A* and *B* respectively.

Proof.

If *A* (or *B*) is involved in any linkage, from Assumption 6.1 we know it is involved in only one linkage, and *A'* (or *B'*) is the linkage node for it. if *A*(or *B*) is not involved in any linkage, *A'* (or *B'*) is a variable node. If both *A'* and *B'* are variable nodes, in step 3 an edge is created for *e*. If one of *A'* and *B'* is a linkage node or both of them are linkage nodes, in the step 5 an edge *A'*? *B'* is created in the J-graph.

Lemma 6.2. Algorithm 1 generates a DAG for the given SLBN.

Proof.

1) It is trivial to see the graph generated by the steps 1, 2 and 3 is a DAG.

2) In the J-graph, if we have edge *A*? *B*, and *A* is a linkage node, let L_A denote the linkage that *A* represents, then from Step 5 we know

   a) if *B* is also a linkage node, let L_B denote the linkage *B* represents. Then because

of the edge $A \rightarrow B$, there exists $X_1 \in src(L_A) \cup dest(L_A)$ and $Y_1 \in src(L_B) \cup dest(L_B)$ such that in a local BN $X_1 \rightarrow Y_1$. And from Theorem 6.1 we know linkages cannot be crossed, so all variables in $src(L_A)$ cannot be the descendant of variables in $src(L_B)$, and all variables in $dest(L_A)$ must be the ancestors or siblings in $dest(L_B)$. Then in the J-graph there does not exist a path from B to A.

b) if B is a variable node, then from the edge $A \rightarrow B$ we know there exists $X_1 \in src(L_A) \cup dest(L_A)$ such that in a local BN $X_1 \rightarrow B$. Suppose both X_1 and B are from the source BN, then from the definition of variable linkage we know that for any $X_i \in src(L_A)$, if $X_i \rightarrow B \rightarrow X_1$, B has to be in $src(L_A)$, which is a contradiction to the fact that B is a variable node in J-graph. So either B and X_i are not in a path or $B \rightarrow X_i$. The same conclusion can be obtained if both X_i and B are from the destination BN. Therefore, we can conclude that in the J-graph there will not exist a path from B to A.

From 1) we can conclude that the generated graph has no cycle including only variable nodes, and from 2) we can conclude that the generated graph has no cycle including any linkage nodes. Therefore we can conclude that the J-graph generated by Algorithm 1 is a DAG. ¦

6.4.2 Constructing the CPT of J-graph

Algorithm 6.2 (Construct the CPTs for J-graph nodes): Given a set of SLBN and the structure of the J-graph is generated by Algorithm 6.1. Let V denote the nodes in the J-graph, V_V denote the variable nodes, and V_L denote the linkage nodes, then $V = V_V \cup V_L$. Also let $p(V_i)$ denote node V_i's parents, Q denote the probability distribution of the

constructed J-graph, and P denote the probability distribution in the linked BNs. Although a node in J-graph may represent more than one BN variables, here we do not make the distinction. V_i and $p(V_i)$ represent both nodes and the variables the node represents.

Assign CPTs to the J-graph's nodes as follows:

1 For any $V_i \in V_V$, if $p(V_i) \subseteq V_V$, from the step 3 of Algorithm 6.1, V_i and $p(V_i)$ must be from the same BN. Assign $Q(V_i|p(V_i))$ the value as the value of the CPT that V_i has in its original BN:

$$Q(V_i \mid p(V_i)) = P(V_i \mid p(V_i)).$$

2 For any $V_i \in V_L$, let L_i denote the linkage that V_i stands for. Then node V_i represents a compound variable that is the combination of $src(L_i)$ and $dest(L_i)$ and has $d(src(L_i))\cdot d(dest(L_i))$ states. A probability distribution $Q(src(L_i), dest(L_i), p(src(L_i)), p(dest(L_i)))$ can be obtained by IPFP on an initially uniform distribution with the following constraints:

$$Q(src(L_i), dest(L_i)) = P_S(src(L_i) \mid dest(L_i))P(dest(L_i)),$$

$$Q(src(L_i), p(src(L_i))) = P(src(L_i), p(src(L_i))),$$

$$Q(dest(L_i), p(dest(L_i))) = P(dest(L_i), p(dest(L_i))), \text{ and}$$

$$Q(p(src(L_i)), p(dest(L_i))) = P(p(src(L_i)))P(p(dest(L_i))),$$

where P denotes the probability distribution from original local BN, and P_S denotes the similarity of L_i. Compute $Q(src(L_i), p(src(L_i)) \mid dest(L_i), p(dest(L_i)))$ and convert it to $P(V|p(V))$.

3 For any $V_i \in V_V$, if some of its parents are linkage nodes, $p_j(V_i) \in p(V_i)$ and $p_j(V_i) \subseteq V_L$. Without losing generality, assume $p_j(V_i)$ contains only one linkage node and let L_j denote the linkage that $p_j(V_i)$ stands for. Linkage nodes $p_j(V_i)$ represents variables $src(L_j)$ and $dest(L_j)$ and has $d(src(L_j))\cdot d(dest(L_j))$ states. Also $p(V_i) - p_j(V_i)+src(L_j)$ represents the parent variables of V_i in its original BN. Then from V_i's original BN obtain $P(V_i|p(V_i)-p_j(V_i)+src(L_j))$. Now initiate a conditional probability as

$$Q(V_i \mid p(V_i)) = \left\{ \frac{1}{d(p(V_i))} \right\}.$$

And then update the conditional probability as

$$\begin{aligned} & Q(V_i \mid p(V_i)) \\ &= Q(V_i \mid p(V_i)) \frac{P(V_i \mid p(V_i) - p_j(X_i), src(L_j))}{Q(V_i \mid p(V_i) - p_j(X_i), src(L_j))}. \end{aligned}$$

and convert $P(V_i \mid p(V_i) - p_j(X_i), src(L_j))$ to $P(V_i|p(V_i))$.

After filling the J-graph with CPTs by Algorithm 6.2, we can see that the probability distribution of the constructed BN includes the probability distributions of the linked BNs. Specifically, in step 1 the variable nodes has CPTs

$$Q(V_i \mid p(V_i)) = P(V_i \mid p(V_i));$$

in step 2 the linkage nodes has CPTS

$$Q(src(L_i) \mid p(src(L_i))) = P(src(L_i) \mid p(src(L_i))) \text{ and}$$

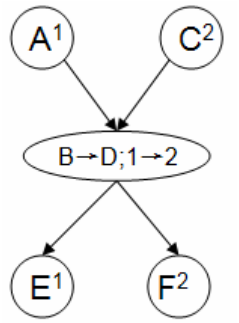
$$Q(dest(L_i) \mid p(dest(L_i))) = P(dest(L_i) \mid p(dest(L_i)));$$

and in step 3 the variable nodes has CPTs

$$Q(V_i \mid p(V_i) - p_j(X_i), src(L_j)) = P(V_i \mid p(V_i) - p_j(X_i), src(L_j)).$$

Therefore, for each variable X_i, no matter it is represented by variable nodes or linkage nodes, its conditional probabilities to its original parents remain unchanged. Specifically, this is stated as

$$Q(X_i \mid p(X_i)) = P(X_i \mid p(X_i)).$$



B	D	F	
		True	False
True	True	0.8	0.2
True	False	0.05	0.95
False	True	0.8	0.2
False	False	0.05	0.95

B	D	C	
		True	False
True	True	0.35	0.65
True	False	0.35	0.65
False	True	0.8	0.2

A	C	B/D			
		True/True	True/False	False/True	False/False
True	True	0	0.9392	0.0608	0
True	False	0	0.7825	0.2175	0
False	True	0	0.2433	0.7567	0
False	False	0	0.0700	0.9300	0

Figure 6.4 The J-graph for the example in Figure 4.9

Figure 6.4 depicts an example of J-graph for the linked BNs in Figure 4.9(a).

Theorem 6.2: Algorithm 6.1 and Algorithm 6.2 generate a J-graph for given SLBN.

Proof.

1 From Lemma 6.2 we know the graph structure generated by Algorithm 6.1 is a DAG.

2 Algorithm 6.1 clearly generated the variable nodes and linkage nodes as the definition requires.

3 From Lemma 6.1 we know every edge of the local BN has a correct representation in the J-graph.

4 From Algorithm 6.2 we can see

a) for every variable *A* in every local BN, its representation in the J-graph has the same conditional probability to its parents;

b) the IPFP process in step 2 of Algorithm 6.2 respects the conditional probability that quantifies the given variable linkage.

Therefore, from the definition of J-graph we can see that Algorithm 6.1 and Algorithm 6.2 generate the J-graph for given SLBN ¦

6.4.3 Validate a SLBN by its J-graph

Till now, we have claims quite some restrictions on SLBN, and some of them, like Assumption 6.2 and 6.3, cannot be easily verified. J-graph is the tool that can help people to valid an SLBN. Suppose a J-graph has been generated by Algorithm 6.1 and 6.2, then

we can check if the given SLBN satisfies all the restrictions from assumptions and definitions by observing its J-graph:

1 If the J-graph for the given SLBN can be obtained by Algorithm 6.1 and 6.2, then in the step 2 of Algorithm 6.2, a joint probability distribution $Q(src(L_i), dest(L_i), p(src(L_i)), p(dest(L_i)))$ has been constructed by IPFP for every variable linkage L_i. This Q can identify that the linkage is consistent with the linked variables. As all linkages can be identified as consistent with the linked variables, all linked BNs are consistent. Therefore, *if a J-graph can be obtained by Algorithm 6.1 and 6.2, the SLBN satisfies the Assumption 4.1*.

2 Assumption 6.1 can be easily checked. It can also be checked in J-graph since if a variable is involved by more then one linkage, then there would be more than one linkage node representing it.

3 Assumption 6.2 can be checked by reviewing the J-graph and see if the J-graph represents a reasonable model about the domain. However, this check still has to be conducted by some domain expert.

4 *If the structure of J-graph is not a DAG, then one or more restrictions in Assumption 6.3 and the definition of variable linkage are violated*: suppose we have a cycle path in the generated J-graph, then the cycle must have a linkage node. We can represent the cycle by $path(A, B, C, A)$, where A is a linkage node.

   a) If B and C are from the same BN, then the restrictions in the definition of variable linkage are violated because B and C are not included as the source variables of the linkage for A.

b) If *B* and *C* are from different BNs, then there must be another linkage node between *B* and *C* so that variables from different BNs can form a path in J-graph. So the path changes to *path*(*A*, *B*, *A'*, *C*, *A*), where *A'* is also a linkage node. Clearly, the linkage of *A* and the linkage of *A'* are crossed. From the proof of Theorem 6.1 we know if two linkages are crossed, then the restrictions in either Assumption 6.3 or the definition of variable linkage are violated.

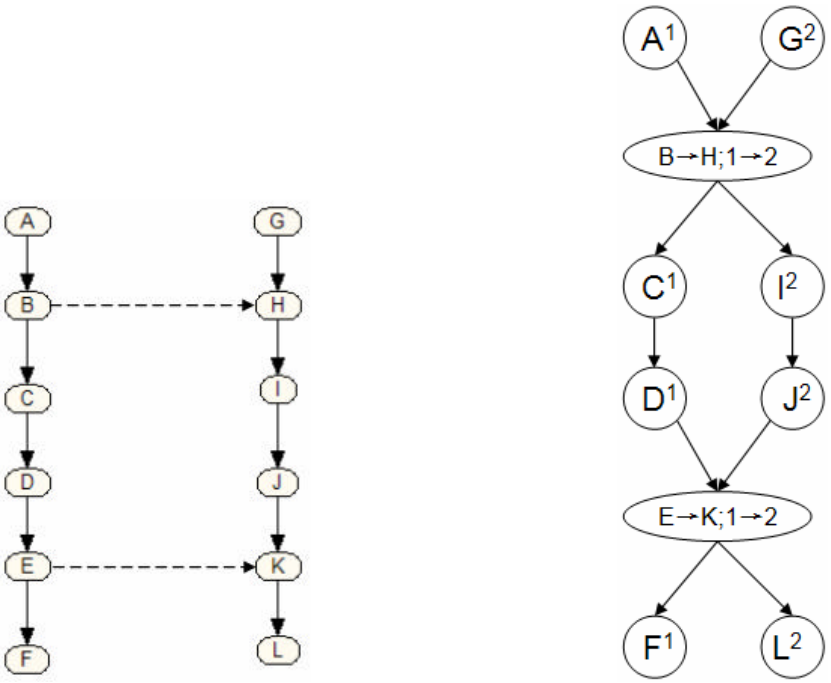
6.5 Inference on J-graph

As we know, each BN represents a probability distribution, and intuitively the ideal approach to conduct inference on SLBN could be to construct a global distribution using the probability distributions of each linked BNs and conduct inferences on this global distribution. Specifically, suppose N_1, N_2, …, N_n are a set of SLBN with linkages L_1, L_2, …, L_m, and they are consistent. N_i encodes a probability distribution P_i on variables X_i. Then ideally we have a joint distribution $Q(X_1, X_2, \ldots, X_n)$, such that $Q(X_i) = P_i(X_i)$. Then the issue is what kind of inference should be conducted on this distribution.

Although J-graph suggests the probability distribution Q exists for the given SLBN by the following formula:

$$Q(X_1, X_2, ..., X_n) = \prod_i Q(X_i \mid p(X_i)),$$

its probability distribution is not suitable for evidential inference to be conducted on using Bayes rule because the variable linkages carry no probabilistic dependencies but semantic similarities.



(a) Two linked BNs (b) the J-graph of (a)

Figure 6.5 An SLBN with two variable linkages

Figure 6.5(a) depicts two linked BNs with two variable linkages L_H^B and L_K^E and Figure 6.5(b) depicts the J-graph for the linked BNs in Figure 6.5(a). Suppose variable A is instantiated to a_1, if we want to update $Q(A, \ldots, L)$ using Bayes rule, we can treat the J-graph as a Bayesian network and instantiate $A = a_1$. From Figure 6.5(b) we can see that if the J-graph is a BN, then

$$Q(F \mid A) = \sum_{D,J} Q(F \mid D, J) Q(D, J \mid A),$$

which indicates a probabilistic influence from J to F. Clearly a probabilistic influence from J to F has a reverse direction than given linkages. Also according such inference method, $Q(K \mid E) \neq Q(K \mid E, D, J)$, and so $Q(K \mid E) \neq Q(K \mid E, A)$, which shows L_K^E is not ensured to be invariant with respect to the evidence $A = a_1$.

As J-graph joins all available information in the linked BNs and the variable linkages, the inference on J-graph should be the inference on the linked BNs. In Chapter 5 we have provided an inference method for SLBN, and here we will justify this method with the help of J-graph. The belief update on the destination variables has been well justified in Section 5.1.1 using Figure 5.2, and here we will justify the belief update methods described in Section 5.1.2 by clarifying the probabilistic dependencies between variables across linkages in J-graph.

Figure 6.6 depicts a J-graph in which a linkage is from *B* to *B'*, variables *A*, *B* and *C* are from the source BN, and *A'*, *B'*, and *C'* are from the destination BN. Because the edges in J-graph also represent causal influences as BN's edges, then their d-separation is also the same as stated in BN[6]:

1 If *B* and *D* are not instantiated, *A* and *A'* are d-separated by *B*/*B'*;

2 If *B* is instantiated, *A* and *D'* are d-separated by *B*/*B'*;

3 If *B* is instantiated, *D* and *A'* are d-separated by *B*/*B'*;

4 If *B* is instantiated, *D* and *D'* are d-separated by *B*/*B'*.

Also, if two variables are not d-separated, they are d-connected. As probabilistic influences can only be propagated along the direction of the variable linkage, if a variable *A* in the source BN and a variable *A'* in the destination BN are d-connected, then the belief change on *A* can change the belief on *A'*, but the belief change on *A'* cannot change the belief on *A*.

[6] Please note that only the variables in the source BN can be instantiated as probabilistic influences can only be propagated from the source BN to the destination BN.

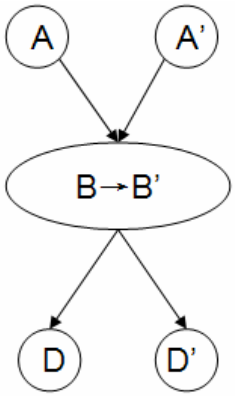


Figure 6.6 The probabilistic dependencies between variables across linkages

Therefore, as the d-connection properties of variables across the linkages are clarified, we can tell how the variables in the destination BN should be updated:

1 When hard evidences update *B* from top (Case 1 in Section 5.1.2), *D'* should be influenced but *A'* should not be updated as it is d-separated from *A* by the linkage node.

2 When hard evidences update *B* from bottom (Case 2 in Section 5.1.2), both *A'* and *D'* should be updated.

3 When hard evidences update B from both top and bottom (Case 3 in Section 5.1.2), *A'* and *A* are d-connected by the evidences from *B*'s bottom, and hence both *A'* and *D'* should be updated.

Apparently, 1, 2, and 3 are exactly what we present in Section 5.1.2.

6.6 Summary

In this chapter we report our investigation on how to model the global knowledge of an SLBN. It would be ideal that the global knowledge of an SLBN could be in the form of a

BN, but unfortunately, the information provided by the linked BNs and the variable linkages is insufficient for us to identify a BN who encodes the global knowledge. So we try to use another model to present the joint knowledge of an SLBN. This model, named J-graph, joins together all available information from the linked BNs and the variable linkages to a probabilistic graphical model. To support this join process, we claim three assumptions on the variable linkages and the linked BNs. These assumptions reduce the complexity of the linkages and require the linked BNs follow the same causal modeling. Then we propose the definition of J-graph and a way to construct a J-graph for the given SLBN. As J-graph needs the support from the restrictions of the assumptions and the definitions, it can also be used to check if these restrictions are met. Finally, we use J-graph to justify the inference methods provided in Chapter 5.

Till now, we have completely reported the SLBN as a principled framework that supports distributed uncertainty reasoning. In the next chapter we will apply SLBN to discover complex concept mappings between semantic web ontologies.

7 Concept Mapping Utilizing SLBN

Uncertainty becomes more prevalent in concept mapping between two ontologies where it is often the case that a concept defined in one ontology can only find partial matches to one or more concepts in another ontology. In other words, they are semantically similar but not identical. Semantic similarities between concepts are difficult, if not impossible to be represented logically, but can easily be represented probabilistically. Similar ideas have motivated recent development of ontology mapping taking probabilistic approaches (GLUE [13], CAIMAN [24], OntoMapper [38], and OMEN [27], see [31] for a survey of existing approaches to ontology mapping, including those based on logical translation, syntactical and linguistic analysis). However, these existing approaches fail to completely address uncertainty in mapping. For example, GLUE captures similarity between two concepts onto1:A and onto2:B by joint probability distribution $P(A, B)$ obtained by text classification of exemplars (semantically relevant text documents) to each concept. Then onto1:A is mapped to onto2:C whose similarity to onto1:A, measured by, say their Jaccard coefficients (computed from the joint distribution) [39], passes a threshold and is the highest among all concepts in onto2. Here, onto1:A is taken as (semantically) equivalent to onto2:C, the degree of similarity (and dissimilarity) between them will not be considered in future reasoning (e.g., subsumption within onto2 after the mapping). Also ignored are the other concepts that are also similar to onto1:A, such as the ancestors of onto2.C (albeit at smaller degree).

The work reported in this chapter utilizes SLBN along with *BayesOWL* to make full use of the probabilistic information concerning relationships between concepts both within and across BNs. As depicted in Figure 7.1 below, this ontology mapping framework consists of three components: 1) a text classification based *learner* to learn from web data the probabilistic ontological information within individual ontologies and between concepts in two different ontologies; 2) a *BayesOWL* module to translate given ontologies (together with the learned uncertain information) into BNs; and 3) an *SLBN module* which takes a set of learned raw similarities as input, creates variable linkages and finds new mappings between concepts from two different ontologies based on evidential reasoning across two BNs.

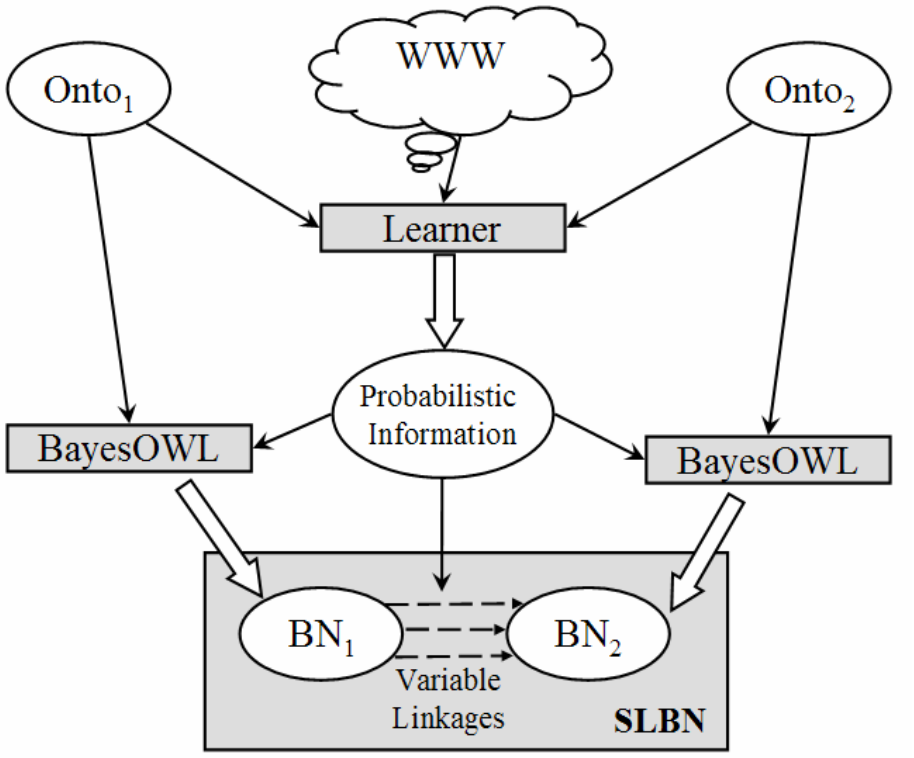


Figure 7.1 Apply SLBN to ontology mapping

In Section 7.1 we will briefly introduce the learner model. The detail about BayesOWL can be found at Section 2.4.1. In Section 7.2 we will present our experiments on two real world ontologies.

7.1 Learning Probabilities from Web Data

Learning the probabilities for semantic similarity between concepts in two ontologies is straightforward, assuming we have sufficient exemplars of good quality associated with each concept. First, we can build a model (classifier) for each concept in Ontology 1 according to the statistical information in that the concept's exemplars using a text classifier such as Rainbow[7] or Bayesian text classifier dbacl[8]. Then concepts in Ontology 2 are classified into classes of Ontology 1 by feeding their respective exemplars into the models of Ontology 1 to obtain a set of probabilistic scores. These scores showing the inter-concept similarity in a probability form. Concepts in Ontology 1 can be classified in the same way into classes of Ontology 2. This cross-classification process (Figure 7.2) helps find a set of raw mappings between Ontology 1 and Ontology 2. Similarly, we can obtain prior or conditional probabilities related to concepts in a single ontology through self-classification with the models learned for that ontology.

In the experiments, for each concept A, we search the web to obtain two sets of exemplars: U^{A+} containing exemplars that support (or positively related to) A; and U^{A-}, containing exemplars that support the negation of (or negatively related to) A. Exemplars in U^{A+} are obtained by searching the web for pages that contain the name of A and all

[7] http://www-2.cs.cmu.edu/~mccallum/bow/rainbow
[8] http://www.lbreyer.com/

names of *A*'s ancestors on the taxonomy, while that for U^{A-} are obtained by search pages that contain all names of *A*'s ancestors but not the name of *A*.

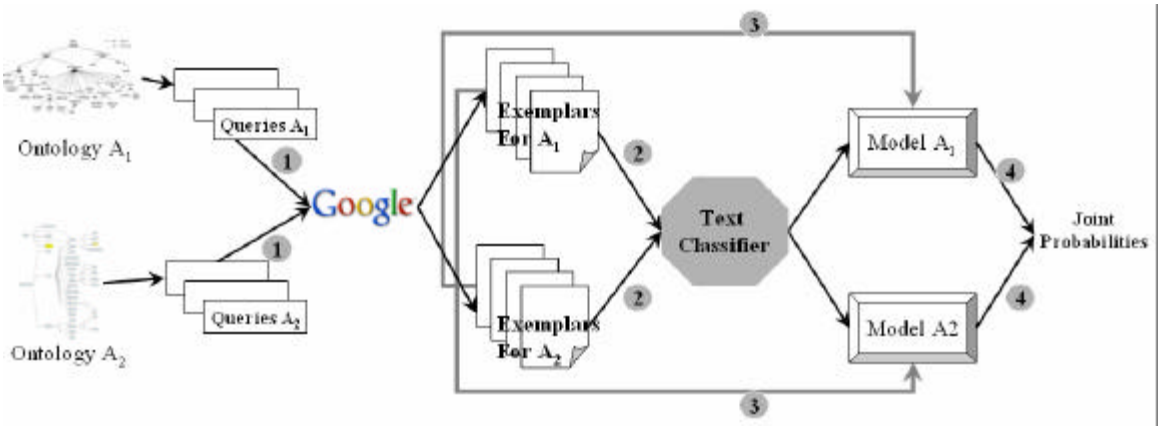


Figure 7.2 Cross-classification using Text Classifiers on Web Data

With all these documents, we can obtain joint probabilities of *A* and *B* by text classification, similar to what is done in GLUE [13]: applying the classifiers of concepts *A* and *B* to all text documents in *U*, where *U* denotes the union of *A*'s exemplars and *B*'s exemplars, and classify them into four categories: U^{A+B+}, U^{A+B-}, U^{A-B+}, and U^{A-B-}. Then the joint probabilities can be obtained by counting the items in each category, e.g., $P(A, B)= |U^{A+B+}| / |U|$.

7.2 Experiments

We have performed computer experiments on two small-scale real-world ontologies. Our goal is to find how good the SLBN's inference could help to find new ontology mappings.

7.2.1 Translating Taxonomies to BNs

We took the Artificial Intelligence sub-domain from ACM Topic Taxonomy[9] and DMOZ[10] (Open Directory) hierarchies and pruned some concepts to form two ontologies, both of which have a single root node *Artificial Intelligence*. All other concepts in the hierarchies are sub categories of AI. These two hierarchies differ in both terminologies and modeling methods. DMOZ categorizes concepts by popularities of web pages to facilitate people's easy access to these pages, while ACM topic hierarchy categorizes concepts from super to sub to structure a classification primarily for academics.

Table 7.1 Statistics of the experiment

Taxonomies	# Nodes	Depth	Total Exemplar size	Avg. Exemplar Size	# Exemplar	Avg. # Exp./node
ACM AI	15	3	19.7 MB	698 KB	24533	1636
DMOZ AI	25	3	29.2 MB	612 KB	35148	1406

For every concept, except the root, we obtained exemplars by querying Google as described in the previous section. The statistics of these web pages is listed in Table 7.1. We used Bayesian text classifier dbacl to create a model for each non-root concept X and obtained the pair-wise conditional probability $P(X \mid Parent(X))$. The root nodes were assigned a prior probability as (0.5, 0.5).

Then, using *BayesOWL*'s translation rules, the two ontologies were translated into two BNs as shown in Figure 7.3.

[9] http://www.acm.org/class/1998/
[10] http://dmoz.org/

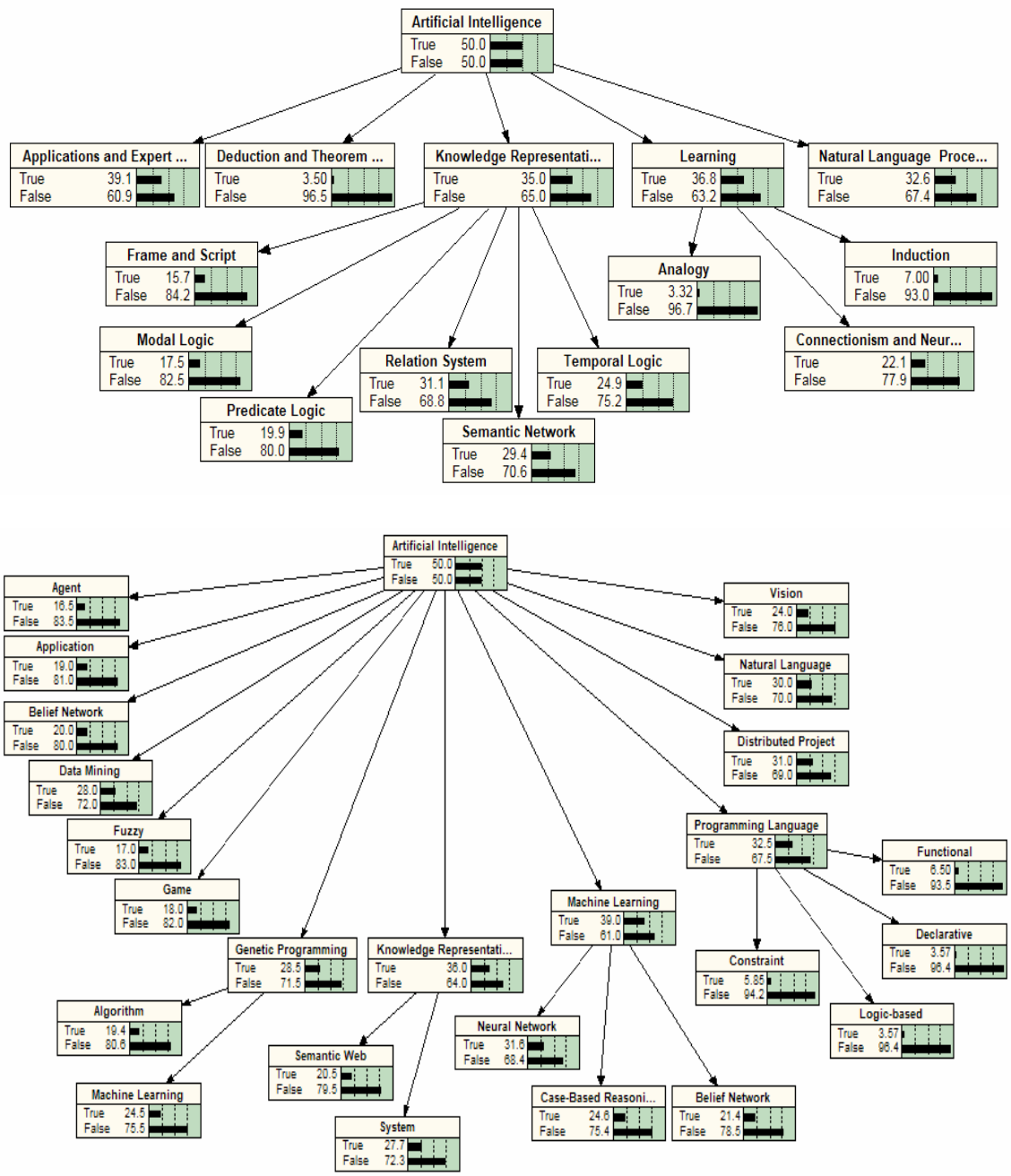


Figure 7.3 Bayesian network for ACM topics' AI sub-domain and DMOZ's AI subdomain

7.2.2 Learning Uncertain Mappings

Raw semantic similarities $P(A|B)$ were computed from P(A, B) for each pair of concepts of the two BNs. The similarity between A and B were measured by their Jaccard coefficient, computed from the joint probability. Table 7.2 lists the five most similar concepts and five most different concepts in the learning result. The top three most similar concepts are actually identical concepts. However, besides these three, another pair of identical concepts is not measured as highly similar. They are /*Learning*/*Connectionism & Neural Net* in ACM topic and /*Machine Learning*/*Neural Network* in DMOZ. Their similarity is only 0.61. We speculate this is because the term "connectionism" is not as popular at present as when ACM topic hierarchy was constructed, and thus is not used along with "*Neural Network*" in most web pages.

Table 7.2 Five most similar concepts and most different concepts in the learning result. The root concept's name is omitted.

ACM topic	DMOZ	Similarity
/Knowledge Representation & Formalism Method	/Knowledge Representation	0.96
/Natural Language Processing	/Natural Language	0.90
/Learning	/Machine Learning	0.88
/Learning	/Knowledge Representation	0.81
/Applications & Expert System	/Knowledge Representation	0.79
……		
/Fuzzy	/Learning/Analog	0.03
/Learning/Induction	/Learning/Game	0.02
/Deduction & Theorem Proving	/Programming Language/Declarative	0.02
/Learning/Induction	/Application	0.01
/Learning/Analogy	/Agent	0.01

7.2.3 Inference with SLBN

Now we can create variable linkages using the learned raw similarities. Clearly, we cannot create variable linkages for every pair of concepts where a learned similarity exists because 1) one variable can only be involved in on linkage (Assumption 6.1), 2)

the variable linkages must be consistent with the linked variables, and 3) not all learned similarities are correct. Clearly, if we can guarantee all the variable linkages we create are true and consistent, then the more variable linkages we have, the more accurate the inference result is.

In this experiment, we created variable linkages between the concepts that are very similar. Specifically, the following linkages are created from the BN for DMOZ, N_{dmoz}, to the BN for ACM, N_{acm}:

$L_1 = < dmoz.kr, acm.krfm, N_{dmoz}, N_{acm}, S_1 >$, where

$$S_1 = P(acm.krfm \mid dmoz.kr) = \begin{pmatrix} 0.9943 & 0.0057 \\ 0.0973 & 0.9027 \end{pmatrix}, \text{ and}$$

$L_2 = < dmoz.nl, acm.nlp, N_{dmoz}, N_{acm}, S_2 >$, where

$$S_2 = P(acm.nlp \mid dmoz.nl) = \begin{pmatrix} 0.9843 & 0.0057 \\ 0.2320 & 0.7680 \end{pmatrix}.$$

Here *acm.krfm* stands for the concept /*Knowledge Representation & Formalism Method* in the ACM ontology, *acm.nl* stands for the concept /*Natural Language Processing* in the ACM ontology, *dmoz.kr* stands for the concept /*Knowledge Representation* in the DMOZ ontology, and *dmoz.nl* stands for the concept /*Natural Language* in the DMOZ ontology. These two pairs of concepts are the most similar concepts between the two ontologies, and their prior distributions in the two BNs are consistent with the variable linkages.

Utilizing SLBN allows us to conduct probabilistic reasoning far beyond finding the best concept match. To illustrate our point, consider the example of finding a description

of DMOZ's /*Knowledge Representation*/*Semantic Web* (*dmoz.sw*) in ACM topics. The two most semantically similar concepts to *dmoz.sw* in ACM are

- /*Knowledge Representation and Formalism Method*/*Relation System* (*acm.rs*) and
- /*Knowledge Representation and Formalism Method*/*Semantic Network* (*acm.sn*)

with the learned joint distributions

$$P(dmoz.sw, acm.rs) = \begin{pmatrix} 0.60 & 0.12 \\ 0.21 & 0.07 \end{pmatrix} \text{ and}$$

$$P(dmoz.sw, acm.sn) = \begin{pmatrix} 0.58 & 0.13 \\ 0.25 & 0.04 \end{pmatrix},$$

and the respective Jaccard coefficients $J(dmoz.sw, acm.rs) = 0.64$, and $J(dmoz.sw, acm.sn) = 0.61$. The similarity between *dmoz.sw* and *acm.krfm*, the super class of *acm.rs* and acm.sn, is even less: $J(dmoz.sw, acm.krfm) = 0.49$. Apparently, there is no ACM concept that is identical to *dmoz.sw*, so it must be described by a composite expression involving multiple ACM concepts. Regular ontology mapping systems, such as GLUE, would fail to find a concept mapping for *dmoz.sw* at this stage.

From the above joint probabilities, we can see that *dmoz.sw* is neither a sub-concept nor a super-concept of *acm.rs* and *acm.sn*, but has a sizable overlap with each of them. From the following joint probabilities

$$P(acm.rs, acm.sn) = \begin{pmatrix} 0.2612 & 0.0498 \\ 0.0323 & 0.6557 \end{pmatrix},$$

we can see that *acm.rs* and *acm.sn* also overlap much with each other.

Using the linkages L_1 and L_2, we can obtain the conditional probabilities of $Q(acm.rs, acm.sn \mid dmoz.sw)$ by the following steps: 1) instantiate *dmoz.sw* to its state *True*; 2) propagate the probabilistic influences from N_{dmoz} to N_{acm} via L_1 and L_2. In this case, the source variable of L_1, *dmoz.kr* is influenced from bottom, and the source variable of L_2, *dmoz.nl*, is influenced from top; 3) update the belief of N_{acm} to obtain $Q_{acm}(acm.rs, acm.sn, \mid dmoz.sw = True)$. Then we calculate $Q_{acm}(acm.rs, acm.sn, \mid dmoz.sw = False)$ analogously with *dmoz.sw* instantiated to its state *False*.

From $Q_{acm}(acm.rs, acm.sn \mid dmoz.sw)$, we find that

$$Q_{acm}(acm.rs = True \vee acm.sn = True \mid dmoz.sw = True) = 0.9646.$$

And we can further calculate $Q(acm.rs, acm.sn, dmoz.sw)$ using prior of $P_{dmoz}(dmoz.sw)$ in N_{dmoz} and estimate the similarities (the Jaccard coefficient) of *dmoz.sw* and $acm.rs \vee acm.sn$:

$$J(dmoz.sw, acm.rs \vee acm.sn) = 0.7250.$$

From the above results we can conclude that the union of *acm.rs* and *acm.sn* is a good mapping of *dmoz.sw*.

8 Conclusion and Discussions

The Bayesian network is an effective representation of probabilistic information using graphical structures. In real world applications, BNs can be easily used to model domain concepts and the probabilistic dependencies between domain concepts by variables with finite states and conditional probabilities between variables. However, inference on BN is not so straightforward. To utilize the structure of the network for computational efficiency, BN localizes the probabilistic reasoning to message passing between neighboring variables and avoids directly updating the belief on the full JPD. As a result, BN inference and its justification are very complicated. Inference across multiple BNs is therefore even more complex and challenging.

This thesis reports our efforts on developing a principled framework that supports probabilistic inference across separately developed BNs. We have addressed the issues in connecting different BNs using semantically similar variables and presented a set of solid theoretical results for this framework. First, we proposed the definition of semantic similarity, a similarity that can be quantified and used in probabilistic systems. Then semantic similarities are embodied by variable linkages and quantified by conditional probabilities. SLBN was proposed as a set of variable linkages and their linked BNs. To ensure consistency among SLBN components and to guarantee correct propagations of probabilistic influences in the system, some restrictions have been imposed to the variable linkages and their linked BNs. In the inference process, we take a novice approach of using soft evidence and virtual evidence to represent probabilistic influences from one BN to the other and developed algorithms for belief update with such uncertain

evidences. Finally we developed a new graphical probabilistic model, J-graph, to represent the joint knowledge of a SLBN and use it to justify the inference process of SLBN.

There are several issues that may deserve further exploration in the future work:

1 How to obtain or discover the semantic similar variables? This is a very important issue when we want to apply SLBN to solve a real world problem. This process could be domain specific, and similar concepts can be either discovered by an automatic or supervised machine learning process, or specified by domain experts.

2 How to effectively create variable linkages from the discovered similar concepts? Basically, it will be very useful to have a set of rules, telling how to use variable linkage to connect given similar concepts such that the generated SLBN is correct, effective and efficient.

3 The inference methods discussed in this thesis assumes we are aware of all the linkages and linked BNs, but what if this is not true? A future work could be to deploy the linked BNs into agents and try to minimize the exchanged information during the inference process.

4 J-graph is a start point at which more properties about SLBN can be discussed and analyzed. For example, can we stress more assumptions to SLBN so that inference can be conducted on both directions, and hence change J-graph to a BN? And conversely we may use J-graph to analyze the consequences if some of the assumptions are not held.

5 The soft evidential update algorithms reported in this thesis assume that the given soft evidential findings are consistent. This is a requirement for IPFP to converge. It has been reported by Vomlel [13][14] and Peng and Ding [6] that when constraints are inconsistent, IPFP will not converge but oscillate. How to detect if a set of constraints are inconsistent and how to handle when given constraints are inconsistent are important issues for further research.

6 Another interesting issue about uncertain evidence is the evidence accumulation involving soft evidence. It is well known that virtual evidence, like hard evidence, can accumulate with other types of evidence because it specifies how likely a probabilistic statement is true regardless under what condition the statement is made. Unlike virtual evidence, however, most people believe that the probability distribution given in a soft evidence should not be changed by other observations after it is applied because a soft evidence is a *certain* finding about the evidence variable's distribution. However, this invariance is not absolute, in some situations soft evidences should be changed by other evidences. Possible situations include what-if analysis and decision supporting systems (e.g., influence diagram) where hypothetical hard evidences or uncertain evidences may be posted after observed evidences have been entered.

With the research in these areas we hope to make SLBN a step closer to the practical applications.

9 References and Bibliography

[1] Ash, R.B. 1970. Basic Probability Theory. John Wiley & Sons, Inc.

[2] Bloemeke, M. 1998. Agent Encapsulated Bayesian Networks. PhD thesis, Department of Computer Science, University of South Carolina.

[3] Bloemeke, M. 2002. Agent-Encapsulated Bayesian Networks and The Rumor Problem. Technical Reports, TR 2002-006, Depart of Computer Science, University of South Carolina.

[4] Bock, H. H. 1989. A Conditional Iterative Proportional Fitting (CIPF) Algorithm with Applications in the Statistical Analysis of Discrete Spatial Data. Bull. ISI, Contributed Papers of 47th Session in Paris, 1: 141-142.

[5] Chan, H. and Darwiche, A.; 2003. Revisiting the Problem of Belief Revision with Uncertain Evidence, in Proceeding of 18th International Joint Conference on Artificial Intelligence.

[6] Csisz'ar, I. 1975. I-divergence geometry of probability distributions and minimization problems. Ann. Prob., 3(1):146--158.

[7] Cooper, G. 1990. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Network. In Artificial Intelligence, 42: 393-347.

[8] Costa, P.; Laskey, K. B.; and Laskey K. J.; 2005. PR-OWL: A Bayesian Framework for the Semantic Web. In Proceedings of the first workshop on Uncertainty reasoning for the Semantic Web, Galway, Ireland.

[9] Cramer, E. 2000. Probability Measures with Given Marginals and Conditionals: I-projections and Conditional Iterative Proportional Fitting, Statistics and Decisions, vol. 18, pp. 311-329.

[10] Deming, W.; Stephan, F. 1940. On a Least Square Adjustment of a Sampled Frequency Table when the Expected Marginal Totals are Known, Ann. Math. Statist. 11, pp. 427-444.

[11] Ding, Z.; Peng, Y.; Pan, R. 2004. A Bayesian Approach to Uncertainty Modeling in OWL Ontology. In Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA2004).

[12] Ding, Z.; Peng, Y.; and Pan, R.; 2005. BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. In Soft Computing in Ontologies and Semantic Web, Springer-Verlag.

[13] Doan, A.; Madhavan, J.; Domingos, P.; and Halevy, A. 2004. Ontology Matching: A Machine Learning Approach. Handbook on Ontologies in Information Systems, S. Staab and R. Studer (eds.), Springer-Velag, 2004. Invited paper. P397-416.

[14] Dou, D.; McDermott, D.; Qi, P. 2002. Ontology Translation by Ontology Merging and Automated Reasoning. In Proc. of EKAW Workshop on Ontologies for Multi-Agent Systems.

[15] Fukushige, Y. October 2004. Representing Probabilistic Knowledge in the Semantic Web. Position paper for the W3C Workshop on Semantic Web for Life Sciences. Cambridge, MA, USA.

[16] Grove A.; Halpern, J. 1997. Probability update: Conditioning vs. cross-entropy. In Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97), pages 208--214, Providence, RI.

[17] Holi, M.; Hyvönen, E. 2004. Probabilistic Information Retrieval Based on Conceptual Overlap in Semantic Web Ontologies, in Proceedings of the 11th Finnish AI Conference, Web Intelligence, vol. 2, Finnish AI Society, Finland.

[18] Jeffery, R. 1983. The logic of Decisions 2nd Edition, University of Chicago Press.

[19] Jensen F. 1994. An Introduction to Bayesian Networks. Aalborg University, February.

[20] Kim, Y.; Valtorta M. 2004. A Prototypical System for Soft Evidential Update. Applied Intelligence, 21, 1 (July-August 2004), 81-97.

[21] Koller, D.; Pfeffer, A. 1997. Object-oriented Bayesian networks. In Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI), Providence, Rhode Island, pages 302—313.

[22] Kruithof, R. Telefoonverkeersrekening, 1937. De Ingenieur 52, E15-E25.

[23] Kyburg, H.E. 1987. Bayesian and non-Bayesian evidential updating. Artificial Intelligence, 31:271--293.

[24] Lacher, M.; and Groh, G. 2001. Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. In Proceedings of the 14th International FLAIRS Conference. Key West, FL, USA.

[25] Laskey, K.B.; Mahoney, S.M; Wright, E. 2001. Hypothesis management in situation-specific network construction. In Proceeding of the seventeenth Annual Conference on Uncertainty in Artificial intelligence (UAI-01), pages 301-309.

[26] Lauritzen, S.L.; Spiegelhalter, D.J. 1988. Local Computation with Probabilities in Graphic Structures and Their Applications in Expert Systems. In J. Royal Statistical Soc. Series B,50(2): 157-224.

[27] Mitra, P.; Noy, N. F.; and Jaiswal, A. R. 2004. OMEN: A Probabilistic Ontology Mapping Tool. In Workshop on Meaning Coordination and Negotiation at the Third International Conference on the Semantic We*b (ISWC-2004)*. Hisroshima, Japan.

[28] Murphy, K.; Weiss, Y.; Jordan, M. 1999. Loopy belief propagation for approximate inference: an empirical study, UAI.

[29] Neapolitan, R.E. 1990. Probabilistic Reasoning in Expert Systems: Theory and Algorithms. John Wiley and Sons, New York, NY.

[30] Noy, N. F.; and Musen, M. A.; 2000. PROMPT: Algorithm and Tool for Automated ontology Merging and Alignment. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000), Austin, TX.

[31] Noy, N. 2004. Semantic integration: A survey of ontology-based approaches. SIGMOD Record.

[32] Paris J.; Vencovska, A. 1992. A method for updating that justifies minimum cross entropy. International Journal of Approximate Reasoning, 7:1--18.

[33] Pearl, J. 1986. Fusion, Propagation, and Structuring in Belief Networks. In Artificial Intelligence, 29: 241-248.

[34] Pearl, J. 1987. Evidential Reasoning Using Stochastic Simulation of Causal Models. In Artificial Intelligence, 32: 245-257.

[35] Pearl, J. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, San Mateo, CA.

[36] Pearl, J. 1990. Jeffery's rule, passage of experience, and neo-Bayesianism. In H.E. et al. Kyburg, Jr., editor, Knowledge Representation and Defeasible Reasoning, pages 245-265. Kluwer Academic Publishers.

[37] Peng, Y. and Ding Z.; 2005. Modifying Bayesian Networks by Probability Constraints, in Proceedings of 21st Conference on Uncertainty in Artificial Intelligence, Edinburgh, Scotland.

[38] Prasad, S., Peng, Y., and Finin, T. 2002. A Tool For Mapping Between Two Ontologies, International Semantic Web Conference (ISWC02), (poster), Sardinia, Italy.

[39] Rijsbergen, V. 1979. Information Retrieval. London:Butterworths, 2nd Edition.

[40] Valtorta, M.; Kim, Y.; Vomlel, J. 2002. Soft evidential update for probabilistic multiagent systems, International Journal Approximate Reasoning 29(1) 71-106.

[41] Vomlel J. 1999. Methods of Probabilistic Knowledge Integration. PhD thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University.

[42] Xiang Y.; Poole D.; M. P. Beddoes. 1993. Multiply Sectioned Bayesian Networks and Junction Forests for Large Knowledge Based Systems, Computational Intelligence, Vol.9, No.2, 171-220.

[43] Xiang, Y.; Lesser, V. 2000. Justifying multiply sectioned Bayesian networks. In Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000), Boston, MA.

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

[45] http://www.auai.org/ Association for Uncertainty in Artificial Intelligence Homepage.

[46] http://www.w3.org/2001/sw/ Semantic Web Homepage.

[47] http://www.w3.org/RDF/ W3C RDF Homepage.

[48] http://www.w3.org/TR/rdf-syntax-grammar/ RDF/XML Syntax Specification

[49] http://www.w3.org/TR/rdf-mt/ RDF Semantics

[50] http://www.w3.org/TR/rdf-schema/ W3C RDF Schema Specification

[51] http://cs-www.cs.yale.edu/homes/dvm/daml/ Yale OntoMerge Homepage, 2002.

[52] http://www.w3.org/TR/owl-ref/ OWL Web Ontology Language Reference

[53] http://jena.sourceforge.net Jena Homepage

[54] http://www.cs.yale.edu/homes/dvm/daml/pddl_daml_translator1.html PDDAML Homepage

[55] http://www.ksl.stanford.edu/software/JTP/ SRI/KSL JTP Homepage

[56] http://www-2.cs.cmu.edu/~mccallum/bow/ Bow Toolkit Homepage