A Junction Tree Propagation Algorithm for Bayesian Networks with Second-Order Uncertainties

Maurizio Borsotto & Weihong Zhang
GCAS Incorporated
1531 Grand Avenue
San Marcos, CA 92078

Emir Kapanci & Avi Pfeffer
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138

Christopher Crick
Department of Computer Science
Yale University
New Haven, CT 06511

Abstract

Bayesian networks (BNs) have been widely used as a model for knowledge representation and probabilistic inferences. However, the single probability representation of conditional dependencies has been proven to be over-constrained in realistic applications. Many efforts have proposed to represent the dependencies using probability intervals instead of single probabilities. In this paper, we move one step further and adopt a probability distribution schema. This results in a higher order representation of uncertainties in a BN. We formulate probabilistic inferences in this context and then propose a mean/covariance propagation algorithm based on the well-known junction tree propagation for standard BNs [1]. For algorithm validation, we develop a two-layered Markov likelihood weighting approach that handles high-order uncertainties and provides "ground-truth" solutions to inferences, albeit very slowly. Our experiments show that the mean/covariance propagation algorithm can efficiently produce high-quality solutions that compare favorably to results obtained through painstaking sampling.

1 INTRODUCTION

A Bayesian Network (BN) is a graphical representation of a joint probability distribution over a set of variables (also called *nodes*) [14]. A common critism of the BN approach is that it requires a single probability representation for the conditional dependencies specified by the network [13]. For instance, if the sky is cloudy, then the probability that it will rain may be (for example) 0.8. However, in knowledge representation, this requirement is inadequate to represent the complications in the real world. When an expert comes up with a single number 0.8, he might indicate that it is a simplification of a range $[0.7, 0.9]$, or that it is a simplification of the fact that it is a probability distribution with 0.8 as mean and 0.1 as standard deviation. In addition, oftentimes different experts have varying assessments of the probabilities for the same application scenario. One expert can conclude that the aforementioned probability is 0.8, while another can feel that it is too large and would suggest 0.7. Single probability representation is insufficient for reconciling discrepancies among a team of experts.

To extend the expressiveness of standard BNs, a concept of *higher-order uncertainty* has been introduced. Semantically, a higher-order uncertainty specifies additional levels of uncertainty beyond the probability distribution representation. For distinction, the uncertainties that can be represented by the conventional way is called *first-order uncertainty* (FOU). For example, given the "cloudy" status, the probabilities for "rain" and "not-rain" are single values and sum up to 1.0. One higher-order representation is to use *interval representation*. This approach has been studied by the imprecise probability and credal network communities [18, 5]. Under this representation, given the "cloudy" status, both probabilities for "rain" and "not-rain" are intervals. Following the interval concept, several interval propagation algorithms have been developed to propagate high-order uncertainties [17, 3, 7, 5, 6]. One limitation of interval propagation is that even in simple situations it is not highly informative [2]. The next section will illustrate this with an example. In addition, even when an algorithm starts with very narrow intervals, those generated by the algorithm oftentimes become too broad to be useful.

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

In this paper, we propose a further extension in representing conditional dependencies. Instead of an interval, we represent them with a probability distribution. In the cloudy-rain example, given the "cloudy" status, the probabilities for "rain" and "not-rain" are both represented as a distribution. For instance, given that the sky is cloudy, the probability that it rains is a distribution with a mean of 0.8 and standard deviation 0.1, while the probability that it won't rain is a distribution with a mean of 0.2 and standard deviation 0.1. Such a probability distribution is an extension to an interval distribution because an interval can be seen as a uniform distribution bounded by the interval range. The uncertainties represented this way are called *Second-Order Uncertainties* (SOUs) or occasionally *two-layer probabilities*.

In spite of increasing expressiveness with model extensions, it becomes computationally more challenging to handle BNs with SOUs. We will use *FOU-BNs* to denote usual BNs, and *SOU-BNs* to denote BNs with SOUs. A FOU-BN represents a single joint probability distribution. In a FOU-BN, a probability inference $p(Q = q|E = e_0)$ is to calculate the posterior probability that the query variable Q is at one value (or state) q given the evidence $E = e_0$. In general, both Q and E can be nodes' sets. As shown later, a SOU-BN can be seen as a continuous set of probability distributions. Because of this, it is computationally demanding to carry out inferences over SOU-BNs. Indeed, in a SOU-BN, an inference $p(Q = q|E = e_0)$ refers to calculating the probability distribution of $Q = q$ given the evidence $E = e_0$. Therefore, an FOU inference calculates a *probability value* for a state of a variable, while an SOU inference calculates a *probability distribution*. Since conducting inferences in general is a NP-hard problem in FOU-BNs [4, 16], SOU-inferences are at least NP-hard.

Currently, no propagation algorithms for two-layer probabilities are available to conduct inferences over SOU-BNs. In this paper, we adopt the Junction Tree (JT) approach [11] and develop a SOU-JT algorithm to propagate the second-order uncertainties [1]. The standard JT approach clusters nodes into cliques, connects cliques to form a junction tree, propagates the evidences and messages throughout the tree, and finally computes the query result. For SOU-BNs, we design innovative procedures to conduct the message propagation and query answering steps. In addition, to balance the quality and the efficiency requirements in propagating two-layer probability distributions, we choose to propagate their means and covariances. Accordingly, the inference results will be represented by mean and variance of $p(Q = q|E = e_0)$. The propagation algorithm is *approximate* in the sense that it computes only the mean and variance of the event $Q = q$ rather than the exact probability distribution.

To validate the mean/covariance algorithm, we generalize the likelihood weighting algorithm [8, 15] to handle SOUs. The generalized algorithm exploits sampling techniques to generate "ground-truth" results for SOU inferences. We developed the sampling algorithm and used its solutions as our reference. We justified our propagation algorithm by demonstrating that it can efficiently generate high-quality results.

2 RELATED WORK

To increase the expressive power of BNs, researchers have proposed interval and two-layered probabilities for realistic BN applications. Probabilistic interval representation and relevant inferences in BNs have been widely studied in the literature [17, 3, 7, 5, 6]. The idea is to represent the sets of posterior probabilities as polytopes and represent a polytope using its vertices' set. At each propagation step, the algorithm calculates the vertices representing the posteriors of a node. Its major drawback is that the vertices' number grows very fast as the number of the parameters increases. Fagiuoli and Zaffalon proposed a *2U algorithm*, an exact interval propagation method for polytrees [7]. But the 2U algorithm applies only to single-connected BNs with binary variables. Tessem proposed one of the earliest approximate algorithms to propagate interval probabilities [17]. However, in this algorithm speed is achieved at the expenses of accuracy and the interval bounds tend to diverge to $[0, 1]$. In addition to JT, a number of approaches exist for carrying out inferences using mathematic programming. One example is the MultiLinear Program (MLP) technique in credal networks [6]. It formulates an inference as a MLP and then solves the MLP. Its drawback is that the number of constraints grows explosively in the network size, and that a MLP is still a difficult non-linear optimization problem.

As mentioned earlier, one limitation of interval propagation is its loose interval bounds [2]. As interval propagation proceeds, the intervals generated in the algorithm quickly become too broad. In addition, a posterior distribution is poorly represented by an interval. Let us illustrate this using our cloudy-rain BN example that has a link from the node "cloudy" to "rain". It has been shown that, even if the parameters representing conditional dependencies of "cloudy" and "rain" are uniformly distributed in their ranges, the interval representing a posterior probability will not be uniformly distributed [2]. The lack of informative power of interval-based approaches is even more evident when the parameters are not uniformly distributed, as shown in Figure 1. The top two charts represent the distributions of the two received evidences for "cloudy". The posterior distributions for one state of "rain" node is shown in the bottom chart. We see that although the posterior distribution features a large interval (between 0.05 and 0.88), the majority of its distribution is confined in the $[0.05, 0.40]$ interval. In-

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

cidentally, it is well approximated by a Beta distribution.

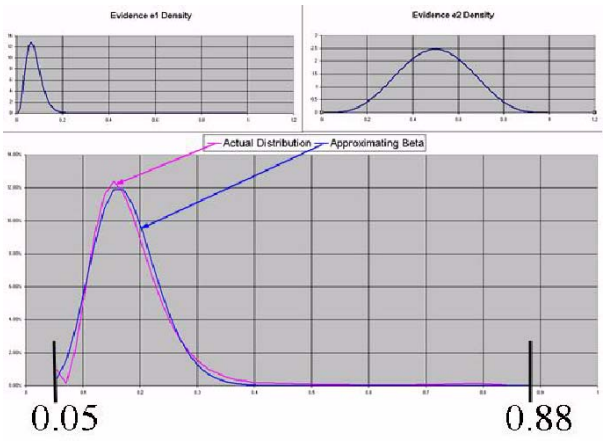

Figure 1. An example for illustrating interval-based propagation and mean/covariance propagation

In spite of the numerous publications on interval probabilities and inferences, there is relatively little work on SOU-BNs that represent conditional dependencies as two-layered probabilities as found in this paper. This is possibly attributable to the computational challenges they pose. Realizing that it might be infeasible to come up with an exact method to probabilistic inferences, we develop the first algorithm to propagate the mean and covariances of nodes through a junction tree [1]. We specifically designed a set of operators for SOUs. These operators are essential for propagating means and covariances. For algorithm validation, we extended the likelihood weighting algorithm to handle SOUs and used it as a reference for our method.

3 BNs AND PROBABILISTIC INFERENCES

A BN is a Directed Acyclic Graph (DAG) that represents a joint probability distribution over a set of variables [14]. In the graph, the nodes denote the random variables in an application, while the links denote the dependencies among them. Quantitatively, the Conditional Probability Table (CPT) of a node encodes the conditional distribution of the node upon the value assignments of its parent nodes. Let the variables' set be $X_1, \ldots, X_n$ and the parent of X_i be π_i. The joint distribution $p(X_1, \ldots, X_n)$ factorizes into: $p(X_1, \ldots, X_n) = \Pi_i p(X_i|\pi_i)$.

We use the notation p_{ijk} to denote the conditional probability $p(X_i = j|\pi_i = k)$ where j is a value of variable X_i and k is a combination of the values of the parents of X_i. Therefore, a BN can be represented by $< \mathcal{N}, \{p_{ijk}\} >$ where $\mathcal{N}$ denotes the DAG qualitative part in terms of nodes and edges, and $\{p_{ijk}\}$ denotes the quantitative part.

An *inference* or *query* $p(Q = q|E = e_0)$ is to calculate the posterior probability distribution of a query variable Q being at its specific value q, given evidence e_0 for node E. In general, Q and E can be sets of nodes.

JT is one of most popular inference algorithms [14]. JT consists of four steps: clustering nodes into cliques, connecting the cliques to form a junction tree, propagating information in the network, and answering a query. A *root* clique is the one with which an inference starts. The core step is message propagation and consists of a message collection phase and a distribution phase. In message propagation, the minimum operational unit is a single message pass. When a message is passed from one clique $\mathbf{X}$ to another clique $\mathbf{Y}$, it is mediated by the sepset $\mathbf{S}$ between the two cliques. For a clique, a *potential* or a *message* is a mapping from value assignments of the nodes to the set $[0, l.0]$. Using the HUGIN architecture [12], a message pass from $\mathbf{X}$ to $\mathbf{Y}$ occurs with two procedures: projection and absorption. The projection procedure saves the current potential and assigns a new one to $\mathbf{S}$:

$$\phi_{\mathbf{S}}^{old} \leftarrow \phi_{\mathbf{S}} \text{ and } \phi_{\mathbf{S}} \leftarrow \sum_{\mathbf{X}\setminus\mathbf{S}} \phi_{\mathbf{X}}. \quad (1)$$

The absorption procedure assigns a new potential to $\mathbf{Y}$, using both the old and the new tables of $\mathbf{S}$:

$$\phi_{\mathbf{Y}} \leftarrow \phi_{\mathbf{Y}} \frac{\phi_{\mathbf{S}}}{\phi_{\mathbf{S}}^{old}}. \quad (2)$$

The query answering step has two procedures. First, the *marginalization* procedure calculates the joint probability of Q and $E = e_0$: $p(Q, E = e_0) = \sum_{\mathbf{X}\setminus\{Q\}} \phi_{\mathbf{X}}$. Second, the normalization procedure calculates the inference result using:

$$p(Q = q|E = e_0) = \frac{p(Q = q, E = e_0)}{\sum_Q p(Q, E = e_0)}. \quad (3)$$

4 SOU-BNs, SOU-INFERENCES AND SOU-POTENTIALS

We introduce SOU-BNs and probabilistic inferences and derive the mean/covariance representation for SOU-BNs. We will discuss probability potentials in the context of SOUs, a fundamental concept in the SOU-JT algorithm.

4.1 SOU-BNs

An SOU-BN is a standard BN $< \mathcal{N}, p_{ijk} >$ where its CPT entries p_{ijk} are probability distributions. To de-

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

note p_{ijk} by a probability density, we introduce the notation $p_{ijk\alpha}$. For any $\alpha \in [0, 1.0]$, the quantity $p_{ijk\alpha}$ is the probability of the event $p(x_i = j|\pi_i = k) = \alpha$, i.e., $p_{ijk\alpha} = p(p(x_i = j|\pi_i = k) = \alpha)$. With $p_{ijk\alpha}$, the probability distribution p_{ijk} is defined by the following set of events $p_{ijk} = \alpha$ and their probabilities:

$$\{(p_{ijk} = \alpha, p_{ijk\alpha})|\forall\alpha \in [0, 1.0]\}. \quad (4)$$

An SOU-BN $< \mathcal{N}, p_{ijk} >$ can be viewed as a probability distribution over a set of FOU-BNs. For any FOU-BN in the set, its CPTs are chosen from the set $\{p_{ijk\alpha}\}$. Moreover, the CPTs must meet the norm constraint, i.e., the probabilities of a node conditioned on a parental assignment must sum up to 1.

This leads to a different query semantics: a FOU-BN query yields a single probability $p(Q = q|E = e_0)$, whereas a SOU-BN query yields a probability distribution of $p(Q = q|E = e_0)$ for a value of the query variable.

4.2 Mean/Covariance Representation of a BN

In the BN definition above, the CPT entries p_{ijk} are represented as distributions, usually as (continuous) probability densities. The representation needs a large amount of memory if the distribution is irregular. To provide a compact representation of a SOU-BN, we define the mean and covariance for all CPTs [1]. Although this representation is an approximation of an SOU-BN, it results in tremendous savings but still preserves the first two-order statistics of a BN, namely its mean and (co)variances. As shown in Figure 1, an approximate Beta distribution, determined using mean and variance estimates, can fit an actual posterior distribution.

Given a CPT entry p_{ijk} in a SOU-BN, we can define its mean and variance as follows:

$$\mu_{ijk} = \int_\alpha p_{ijk\alpha}\alpha d\alpha,$$
$$\sigma^2_{ijk} = \int_\alpha p_{ijk\alpha}(\alpha - \mu_{ijk})^2 d\alpha.$$

The covariances depict the relationship among different values given a node and its parents. The covariance $\sigma^2_{i<j_1,j_2>k}$ is defined as follows:

$$\sigma^2_{i<j_1,j_2>k} = \int_\alpha \int_\beta p_{ij_1k\alpha}p_{ij_2k\beta}(\alpha - \mu_{ij_1k})(\beta - \mu_{ij_2k})d\alpha d\beta.$$

The goal of an inference $p(Q = q|E = e_0)$ is to compute a mean and a variance of the probability distribution of $Q = q$ given evidence e_0.

4.3 SOU POTENTIALS

A potential is defined over a set $\mathbf{X}$ of variables $X_1, \ldots, X_n$. We will use $X_{1:n}$ to represent $(X_1, \ldots, X_n)$ and $x_{1:n}$ to indicate an assignment to it. The potential defined over the set can be denoted by $\phi_{\mathbf{X}}$ or $\phi(X_{1:n})$. In SOU-BNs, a potential has to be interpreted differently from FOU-BNs.

For each value assignment of all variables, the notation $\phi(x_{1:n})$ can be understood as a set of real numbers. Intuitively, one can think of $\phi(x_{1:n})$ as a probability distribution that consists of probability masses $\phi(x_{1:n}; \alpha)$ over any $\alpha \in [0, 1]$. For each $\alpha \in [0, 1]$, $\phi(x_{1:n}; \alpha)$ specifies the probability that the entry $\phi(x_{1:n})$ is equal to α, i.e., $\phi(x_{1:n}; \alpha) = p(\phi(X_{1:n} = x_{1:n}) = \alpha)$.

With this notation, given a potential $\phi(X_{1:n})$, its mean value at $x_{1:n}$ is:

$$\mu_\phi(x_{1:n}) = \int \phi(x_{1:n}; \alpha)\alpha d\alpha$$

and its variance at $x_{1:n}$ is

$$\sigma^2_\phi(x_{1:n}) = \int \phi(x_{1:n}; \alpha)(\alpha - \mu_\phi(x_{1:n}))^2 d\alpha.$$

The covariance for two different instantiations $x_{1:n}$ and $x'_{1:n}$ of the variables' set $\{X_{1:n}\}$ is

$$\sigma^2_\phi(x_{1:n}; x'_{1:n}) = \int_\alpha \int_\beta \frac{\phi(x_{1:n}; \alpha)\phi(x'_{1:n}; \beta)}{(\alpha - \mu_\phi(x_{1:n}))(\beta - \mu_\phi(x'_{1:n}))d\alpha d\beta}$$

where α and β can be any number in $[0, 1]$.

Note that if $x_{1:n} = x'_{1:n}$, the covariance $\sigma^2_\phi(x_{1:n}; x'_{1:n})$ is actually the variance $\sigma^2_\phi(x_{1:n})$. Note also that the CPTs for a variable are examples of potentials. In fact, the CPT potential of a variable is defined over the set of the variable and its parents.

5 JUNCTION TREE PROPAGATION

The proposed SOU Junction Tree (SOU-JT) algorithm parallels the standard JT for FOU-BNs [9]. Just like FOU-JT, SOU-JT proceeds in four steps: construct a junction tree, initialize the tree, conduct message passing via global propagation, and generate the query results [1]. The junction tree construction step is identical to the FOU case. In the rest of this section, we discuss the remaining three steps. For each step, we first discuss the operations and then the operators that implement them. Throughout our discussions, we only introduce the definitions of the operators. We defer our derivations and all proofs to a longer version of this paper [19].

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

5.1 Junction Tree Initialization

The tree initialization sets up the initial potentials for the cliques. In particular, the initial potentials and evidences are assigned to the cliques. Then, within each clique, these potentials are multiplied together to form one single potential. The initialization step proceeds as follows [1]:

1. For each cluster $\mathbf{X}$ and sepset $\mathbf{S}$, set its mean matrix to be 1.0 and its covariance matrix to be 0.0. That is, $\mu_{\phi_\mathbf{X}} \leftarrow 1.0$, and $\sigma^2_{\phi_\mathbf{X}} \leftarrow 0.0$.

2. For each variable X, assign it to a cluster $\mathbf{X}$ that contains the node and all its parents. Such a cluster is *the parental cluster of* X.

3. For each evidence variable E, identify a cluster that contains E and attach a potential λ_E to the clique.

4. For each clique, perform the multiplication operations over the current potential $\phi_\mathbf{X}$, the CPT potentials $p(X|\pi(X))$ for all assigned Xs, and the evidence potential λ_E attached in the clique:

$$\phi_\mathbf{X} \leftarrow \phi_\mathbf{X} \Pi_X p(X|\pi(X)) \Pi_E \lambda_E$$

where Π_X is over all the nodes whose CPTs are attached to the clique and Π_E is over all available evidences. The multiplication operator is defined in the next section.

5.1.1 Multiplication Operator

The multiplication operator acts on a number of potentials f_1,. . .,f_n. Each f_i is represented by a mean matrix and a covariance matrix. The operator returns a potential represented by a mean matrix and a covariance matrix. We assume that f_is have the same domain [1].

We first consider the two-potential multiplication. Let the two potentials be g and h. Let the variables' set be $\{X_{1:n}\}$, the mean matrix and the covariance matrix be $\mu_g(\mu_h)$ and $\sigma^2_g(\sigma^2_h)$, the resulting potential be gh. Then the mean matrix μ_{gh} is [1]:

$$\mu_{gh}(x_{1:n}) = \mu_g(x_{1:n})\mu_h(x_{1:n}) \tag{5}$$

The covariance matrix σ^2_{gh} is [1]:

$$\begin{aligned}\sigma^2_{gh}(x_{1:n}; x'_{1:n}) \quad &= \sigma^2_g(x_{1:n}; x'_{1:n})\sigma^2_h(x_{1:n}; x'_{1:n}) \\ &+\mu_h(x_{1:n})\,\mu_h(x'_{1:n})\sigma^2_g(x_{1:n}; x'_{1:n}) \\ &+\mu_g(x_{1:n})\mu_g(x'_{1:n})\sigma^2_h(x_{1:n}; x'_{1:n}).\end{aligned} \tag{6}$$

Equations (5) holds because the various CPTs in a BN are independent. Indeed, Equation (5) can be verified using a Taylor expansion. Equation (6) can be derived by expanding the covariance of gh at $(x_{1:n}; x'_{1:n})$ considering that g and h are independent because each CPT appears exactly in one clique.

Additionally, we have proven the following results: (1) If the matrices σ^2_g and σ^2_h are symmetric, so is the covariance matrix σ^2_{gh}; (2) The resulting covariance matrix σ^2_{gh} is independent of the multiplication order among the potentials; and (3) The multiplication operator is associative. The first result can be used to achieve computational savings, while the others can be used in multiplying more than two potentials.

5.1.2 Extension Operator

When the multiplication operator (and the division operator discussed later) acts on two potentials that have different sets of variables, an extension operator is needed to unify their domains. Given a potential $\phi(X_{1:n})$ (with mean μ_ϕ and covariances σ^2_ϕ) and a set of variables $\{Y_{1:m}\}$, *the extension of the potential* $\phi(X_1, \ldots, X_n)$ to include $Y_{1:m}$ is the potential $\phi(X_{1:n}, Y_{1:m})$. For any assignment $(x_{1:n}, y_{1:m})$ to $(X_{1:n}, Y_{1:m})$, the mean is defined as [1]:

$$\mu_\phi(x_{1:n}, y_{1:m}) = \mu_\phi(x_{1:n}).$$

For any $(x_{1:n}, y_{1:m}; x'_{1:n}, y'_{1:m})$, its covariance is defined as [1]:

$$\sigma^2_\phi(x_{1:n}, y_{1:m}; x'_{1:n}, y'_{1:m}) = \sigma^2_\phi(x_{1:n}; x'_{1:n}).$$

By definition, if the extended potential is projected back to the original domain space, the result is the same as the original potential.

5.2 Global Message Propagation

In this step, as in standard JT, we choose a *root clique* to direct the global message passing. The ordering among the cliques are the same as in standard JT. We will concentrate on a single message passing step here.

There are two steps in computing the mean and covariance matrices for the sepset and the clique that receives a message [1]. In projection, Equation (1) is achieved by a sum-out operator. In absorption, Equation (2) is achieved by a combination of a multiplication and division operator. Since the multiplication operator has been defined in the preceding subsection, we will define a sum-out and a division operator.

5.2.1 Sum-out Operator

A sum-out operator acts on a potential and a set of variables. It sums out the variables in the given set and returns a potential defined over a small set of variables. Let the potential

[1] If f_i has a different domain from another f_j, we can apply the *extension operator*, introduced in the next section, to them and then carry out multiplications in an unified domain.

be ϕ and its set of domain variables be $\{X_{1:n}\}$. Without loss of generality, let the node to sum out be X_n. The mean matrix of $\phi(X_{1:n-1})$ is defined to be [1]:

$$\mu_\phi(x_{1:n-1}) = \sum_{x_n} \mu_\phi(x_{1:n-1}, x_n)$$

and the covariance matrix of $\phi(X_{1:n-1})$ is defined to be [1]:

$$\sigma^2_\phi(x_{1:n-1}; x'_{1:n-1}) = \sum_{x_n, x'_n} \sigma^2_\phi(x_{1:n-1}, x_n; x'_{1:n-1}, x'_n).$$

The above operator can be readily extended to sum out multiple nodes. In that case, the variables can be summed out one by one in any order.

5.2.2 Division Operator

The division operator acts on two potentials and returns a quotient potential. Let the two potentials be g and h, and the result be g/h. Given g and h, the mean matrix of g/h is defined to be [1]:

$$\mu_{g/h}(x_{1:n}) = \frac{\mu_g(x_{1:n})}{\mu_h(x_{1:n})} \tag{7}$$

and the covariance matrix $\sigma^2_{g/h}$ at $(x_{1_n}; x'_{1:n})$ of g/h is defined to be [1]:

$$\frac{\sigma^2_g(x_{1:n}, x'_{1:n}) - \mu_{g/h}(x_{1:n})\mu_{g/h}(x'_{1:n})\sigma^2_h(x_{1:n}, x'_{1:n})}{\sigma^2_h(x_{1:n}, x'_{1:n}) + \mu_h(x_{1:n})\mu_h(x'_{1:n})}. \tag{8}$$

Equations (7) and (8) are respectively the inverse of (5) and (6).

5.3 Query Answering

After the clique tree is made consistent via global propagation, we have $\phi_{\mathbf{X}} = p(\mathbf{X}, E = e_0)$ for each cluster (or sepset) $\mathbf{X}$. The query result is obtained in two steps: marginalization and normalization.

- The marginalization step can be accomplished using the sum-out operator, as discussed earlier.
- To normalize, we would use $Y_i = \frac{X_i}{\sum_{i=1}^{n} X_i}$ to replace the node X_i. This yields a set of "normalized" variables. The potential is then defined over the normalized variables and represented by its matrices. These matrices are then exploited to calculate the query result – the mean and the variance of $p(Q = q|E = e_0)$, an approximation to the distribution of $p(Q = q|E = e_0)$. In the following, we define the normalization operator.

5.3.1 Normalization Operator

Given a potential $f(X_{1:N})$, the operator returns the matrices representing a potential over a set of normalized variables $\{Y_n = \frac{X_n}{\sum_i X_i} | n = 1 : N\}$. Let the normalized potential be $\bar{f}$. For each node Y_n, its mean and variances are computed by the following two equations where $\beta = \sum_i \mu_i$ and $\sigma^2_{ij} = \sigma^2_f(x_i, x_j)$ [1].

$$\mu_{\bar{f}}(y_n) = \frac{\mu_n}{\beta} + \frac{1}{2\beta^3}[(2\mu_n - 2\beta)\sigma^2_n + \sum_{i=j\neq n} 2\mu_n\sigma^2_i + (2\mu_n - \beta)(\sum_{i\neq n} \sigma^2_{in} + \sum_{j\neq n} \sigma^2_{nj}) + \sum_{i\neq j\neq n} 2\mu_n\sigma^2_{ij}].$$

$$\sigma^2_{\bar{f}}(y_n) = \frac{1}{\beta^4}[(\beta - \mu_n)^2\sigma^2_n + \mu^2_n \sum_{i\neq n} \sigma^2_i - \mu_n(\beta - \mu_n)(\sum_{i\neq n} \sigma^2_{in} + \sum_{j\neq n} \sigma^2_{nj}) + \mu^2_n \sum_{i\neq j\neq n} \sigma^2_{ij}].$$

The equations are obtained using Taylor expansion of the definition $Y_n = \frac{X_n}{\sum_i X_i}$ at $Y_n = y_n$, as shown in [19].

6 SAMPLING APPROACH FOR SOU-BNs

In this section we generalize the *likelihood weighting* algorithm [8, 15] for BNs to accommodate SOUs. The resulted sampling algorithm, run for a suitably long time, serves as a reference algorithm for validating results obtained with SOU-JT.

The likelihood weighting algorithm relies on random sample generation. For simplicity, imagine a BN with no evidence entered. The algorithm produces a sample drawn from the prior distribution for each node without parents, then samples each child node based on the distribution given already-instantiated parents, until each node in the network has been assigned a state. It performs this process over and over again, remembering the final outcome for each state in each trial. It then averages over all of the trials to estimate the joint probability distribution for each variable in the network.

Our SOU sampling algorithm works similarly, except that for each node, we sample twice: first, to determine the prior probability distribution given the mean and covariance matrices, then to determine an actual outcome given this particular instantiation of the prior distribution. With this, we obtain a first-order estimate of the mean values of the distribution. To generate estimates of the variances, we first generated a sample of prior distributions over all of the nodes in the network, and then for each such set we generated enough samples to characterize the probability distribution. Taking the set of all samples of such prior distributions, we can compute the variance on the probability of any particular node.

The likelihood weighting algorithm used in our referee system gets its name from the treatment of evidence within the network. Any particular trial receives a weight based

on the likelihood of a particular sample given the available evidence. In this way, the sampling process iteratively approaches the true posterior distribution of the network.

7 EXPERIMENTS

7.1 Experimental Setup

The validation process involved several different SOU-BNs, some of which were based on real-world applications and some were randomly generated. The real-world networks included HAILFINDER (a 57-node network for predicting weather in northern Colorado), ALARM (a 37-node network for monitoring patients in intensive care) and WIN95PTS (a 76-node network for diagnosing printer problems). These networks can be found at the repository http://www.cs.huji.ac.il/labs/compbio/Repository/. We made changes to the FOU-BN models to include SOUs by manually augmenting the models with or generating random means and covariances of CPTs. To further evaluate the algorithmic performance, we created two random networks: (1) a very loopy 36-node network arranged in a 6x6 lattice (most nodes having eight neighbors), and (2) a large but sparsely connected graph of 200 nodes. Each node represented between two and six states, and the mean and covariance values of the conditional probability tables were chosen at random.

In each experiment, we used our sampling algorithm to generate approximate referee mean likelihood and variances, then compared those results with the performance of our SOU-JT algorithm. Each node's deviation was calculated using $\frac{|x-\hat{x}|}{x}$, where x represents the value (either mean or variance) generated by the sampling algorithm and $\hat{x}$ represents the value generated by the SOU propagation algorithm. These values were then averaged over all nodes to determine their averages shown in Tables 1, 2 and 3. To accelerate SOU-JT and to ensure numerical stability, we applied zero-compression techniques to the calculations of covariances [10]. These tests were conducted using a HUGIN-based inference engine. More recently, we developed a Lazy Propagation engine [11] featuring several optimization techniques. This has proven so far to be 30 times faster, on average, than the HUGIN-based engine.

7.2 Results

Table 1 shows our best results for networks with no evidence entered. The "Average" row in the table represents the average over all tested real-world BNs available. Many choices of zero-compression threshold were tried, including ones based carefully on the distribution of values within the matrix, but the best results were obtained with a simple constant-value threshold of 0.01. That is, all covariance matrix entries with values less than this are discarded. It can be seen that the results from SOU-JT and the sampling algorithms are close. Hence the SOU-JT algorithm is effective in mean and covariance propagation.

BNs	Maximum mean deviation	Average mean deviation	Average variance deviation
Average	2.14%	0.45%	4.22%
Loopy lattice	3.08	0.64	7.27
Large sparse	0.61	0.19	1.45

Table 1. Best performance

Table 2 shows the performance of the algorithm with random evidence entered. In experiments, 5 and 20 percent of nodes were chosen at random, and an observation distribution over the states in the node imposed. The averages in the tables were taken over all of the nodes which did not have evidence entered. Each table entry has two numbers: the first is for 5 percent case and the second in parentheses is for 20 percent case. We note that adding evidence improves the performance of the networks, though the effect is less noticeable for densely connected networks (Lattice network) than for sparse ones. This is explained by the fact that the values of evidence variables are unaffected by the message passing algorithm, and therefore serve as a firebreak in the propagation of errors.

BNs	Maximum mean deviation	Average mean deviation	Average variance deviation
Average	2.95 (1.83) %	0.61 (0.40) %	3.92 (3.16) %
Loopy lattice	3.41 (3.03)	0.77 (0.79)	8.04 (7.41)
Large sparse	0.40 (0.26)	0.12 (0.09)	1.25 (1.01)

Table 2. Evidence on 5% and 20% of Nodes

We show the timing performance on each network using a 1.83GHz Pentium system in Table 3. We ran the sampling algorithm about four hours on each network. For SOU-JT, in the table, the compilation time is the amount of time it takes to generate the junction tree and do an initial round of message passing to bring the network to a consistent state. The message passing time is the time taken to re-establish consistency after evidence has been entered. We see that most queries can be answered in (nearly) real-time manner from SOU-JT, except those on the highly dense lattice network in which each node has eight neighbors.

This validation process provides solid experimental evidence for the correctness of our SOU-JT algorithm. Under many different network conditions, the means and variances computed by SOU-JT closely match the figures obtained

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

BNs	Compilation	Message passing
HAILFINDER	8.2 sec	1.1 sec
ALARM	5.9	1.0
WIN95PTS	14.6	3.1
Loopy lattice	137.0	46.6
Large sparse	1.1	0.1

Table 3. Running time

from the brute-force simulation. Combining the timing result, we conclude that SOU-JT is able to efficiently provide accurate inference solutions to SOU-BNs.

In addition, the results we obtained suggest that SOU-JT does not suffer from the same problem affecting interval propagation. In interval propagation the a-posteriori intervals tend to diverge towards the $[0, 1]$ interval as inference proceeds along the network. We did not observe this effect so far with the computation of covariances. Further tests are being conducted to verify this behavior.

8 CONCLUSIONS

In this paper, we described a new extension to increase the expressiveness of BNs. We used a second-order probability representation for encoding conditional dependencies among variables. We then discussed probabilistic inferences in this context and their mean/covariance representation. We developed the first clique tree algorithm to propagate mean and covariances for SOU-BNs [1]. To make the algorithm operational, we defined the fundamental operators including extension, multiplication, division, sum-out (marginalization) and normalization for SOU potentials [1]. For algorithm validation, we generalized the likelihood weighting approach to accommodate SOUs and provide ground-truth inference results. Our experimental results showed that the proposed algorithm can efficiently produce high-quality inference results. These appear not to be affected by the divergence problem typical of interval propagation.

Acknowledgements

This material is based on work supported by the Navy Surface Warfare Center, Dahlgren Division under Missile Defense Agency-funded Contracts No. N00178-03-C-3098 and N00178-04-C-3117. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Naval Surface Warfare Center, Dahlgren Division or of the Missile Defense Agency. We are deeply thankful for guidance and suggestions provided by our Navy Technical Monitor, Dr. John Crigler.

References

[1] Patent pending. 2006.

[2] M. Borsotto and C. T. Savell. Decision theory research and development: interval propagation in bayesian network. In *Phase-I FinalReport, MDA-funded SBIR contract N00178-03-C-3098*, 2004.

[3] J. Breese and K. Fertig. Decision making with interval influence diagrams. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence (UAI-91)*, New York, NY, 1991. Elsevier Science Publishing Company, Inc.

[4] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief network. *Artificial Intelligence*, 42:393–405, 1990.

[5] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.

[6] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In *STAIRS*, 2004. Spain.

[7] E. Fagiuoli and M. Zaffalon. 2U: an exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, (1):77–107, 1998.

[8] R. Fung and K. Chang. Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Uncertainty in Artificial Intelligence 5*, pages 209–219, 1989.

[9] C. Huang and A. Darwiche. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[10] F. Jensen and S. K. Andersen. Approximations in Bayesian belief universes for knowledge-based systems. In *the 6th Conference on Uncertainty in Artificial Intelligence*, pages 162–169, 1990.

[11] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.

[12] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4:269–282, 1990.

[13] H. E. Kyburg. Bayesian and non-bayesian evidential updating. *Artificial Intelligence*, 31(3):271–293, 1987.

[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[15] R. D. Shachter and M. A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligence 5*, pages 221–231, 1989.

[16] S. E. Shimoni. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68:399–410, 1994.

[17] B. Tessem. Interval probability propagation. *International Journal of Approximate Reasoning*, 7:95–120, 1992.

[18] P. Walley. *Statistical reasoning with imprecise probabilities*. Chapman and Hall, London, 1991.

[19] W. Zhang, M. Borsotto, E. Kapanci, and C. Crick. Propagating expectations and (co)variances in bayesian network with second-order uncertainty. *GCAS Research Report*, 2005.