Local Propagation in Bayesian networks vs Semi-Join Program in Databases

Dan Wu
School of Computer Science
University of Windsor
Windsor Ontario
Canada N9B 3P4

Michael Wong
Department of Computer Science
University of Regina
Regina Saskatchewan
Canada S4S 0A2

Abstract

In this paper, we study the local propagation method used in Bayesian net for belief updating from a relational database perspective. We point out and prove that the renowned local propagation method is in fact a (generalized) semi-join program used in relational database for answering database queries. This interesting connection implies and suggests that the task of belief updating can be reconsidered as a database query problem.

1. Introduction

Bayesian nets has become so successful and popular in the last decade as a tool for processing uncertainty using probability theory. A Bayesian net consists of a graphical component which is a *directed acyclic graph* (DAG) and a corresponding set of *conditional probability distributions* (CPDs) whose product yields a *joint probability distribution* (JPD) over a problem domain (Pearl 1988). In other words, Bayesian net provides a mechanism to graphically represent a JPD as a factorization of a set of CPDs. The key problem in Bayesian nets is *belief updating*, also called *probabilistic inference*, which simply means computing posterior probability $p(X|e)$, where X is a set of variables and e is the evidence observed (Jensen 1996).

One of the major breakthroughs in the development of Bayesian nets is the discovery of the *local propagation* method for belief updating (Lauritzen & Spiegelhalter 1988) in Bayesian nets. A Bayesian net is normally *moralized* and *triangulated* so that a junction tree is constructed by identifying all the maximal cliques in the triangulated graph and properly arranging them as a tree. The local propagation method is then applied on the resulting junction tree for belief updating; it is by far the dominant method for belief updating and it has received wide acceptance in the Bayesian net community. In fact, the local propagation method has become the de facto standard for belief updating. Since then much effort has been spent on further improving the efficiency of local propagation (Kjaerulff 1990; 1997; Madsen & Jensen 1998).

It is unequivocable that the local propagation technique is vital to the success of belief updating in Bayesian nets. In this paper, we first review the technique of local propagation. We then point out and prove that the local propagation method is in essence a *semi-join* program (Beeri *et al.* 1983) which was developed in early 1980s for answering database queries. This seemingly striking connection actually does not surprise us much because it has been noticed in recent years that there exists an intriguing relationship between Bayesian nets and relational databases (Wong 1997). This connection inspires us to revisit the problem of belief updating from the relational database query processing perspective, and it also implies that the task of belief updating may be reconsidered as a database query problem. The paper is organized as follows. In Sections 2 and 3, we review the local propagation method in Bayesian nets and semi-join programs in relational databases respectively. In Section 4, we reveal and prove that the local propagation technique currently employed in Bayesian nets is indeed a generalized semi-join program in relational database theory. In Section 5, we discuss implication of this connection and conclude the paper.

2. Local Propagation in Bayesian nets

A *Bayesian net* (BN) defined over a set $R = \{a_1, \ldots, a_n\}$ of variables is a tuple $(\mathcal{D}, C)$, where $\mathcal{D}$ is a *directed acyclic graph* (DAG) and $C = \{p(a_i|pa(a_i)) \mid a_i \in R\}$ is a set of CPDs, where $pa(a_i)$ denotes the parents of node a_i in $\mathcal{D}$. The product of the CPDs in C defines the JPD $p(R)$ as follows:

$$p(R) = p(a_1|pa(a_1)) \cdot \ldots \cdot p(a_n|pa(a_n)).$$

Although various methods exist for belief updating, one of the most popular methods is based on a computational scheme called local propagation (Lauritzen & Spiegelhalter 1988; Jensen 1996). Local propagation is *not* directly performed on the DAG of a BN, but on a secondary structure, namely, the *junction tree*. The DAG of a BN is normally transformed through moralization and triangulation into a junction tree on which the local propagation procedure is applied. After the local propagation procedure finishes its execution, a marginal distribution is computed for each node and separator in the junction tree. In the following, we briefly review how local propagation works using a running example.

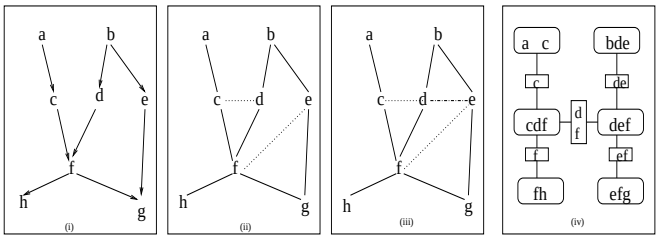

Figure 1: (i) The Asia BN $\mathcal{D}$. (ii) The moral graph $\mathcal{U}_{\mathcal{D}}$ of $\mathcal{D}$ in (i). (iii) The triangulation of $\mathcal{U}_{\mathcal{D}}$ in (ii). (iv) The resulting junction Tree of the BN in (i). (v) The hypertree representation of the junction tree in (iv).

Recall the Asia travel BN defined over $R = \{a,\ b,\ c,\ d,\ e,\ f,\ g\}$ from (Lauritzen & Spiegelhalter 1988). Its DAG $\mathcal{D}$ is shown in Figure 1(i), the CPDs specified for each node in $\mathcal{D}$ are $\{p(a)$,$p(b)$, $p(c|a)$, $p(d|b)$, $p(e|b)$, $p(f|cd)$, $p(g|ef)$, $p(h|f)\}$. The JPD defined by the above CPDs is: $p(R) = p(a) \cdot p(b) \cdot p(c|a) \cdot p(d|b) \cdot p(e|b) \cdot p(f|cd) \cdot p(g|ef) \cdot p(h|f)$. The overall structure of the local propagation technique consists of the following 3 steps (Huang & Darwiche 1996):

(1) *Graphical transformation.* The DAG, for instance, the one in Figure 1 (i), is first transformed to its *moral graph* denoted $\mathcal{U}_{\mathcal{D}}$, by connecting all parents of each node in $\mathcal{D}$ and dropping the directionality of all directed edges. The resulting moral graph $\mathcal{U}_{\mathcal{D}}$ is shown in Figure 1 (ii). This moral graph $\mathcal{U}_{\mathcal{D}}$ is then *triangulated* by connecting two nonadjacent nodes in every cycle of length four or greater in $\mathcal{U}_{\mathcal{D}}$. In our example, one possible triangulation is to connect nodes d and e, the resulting triangulated graph is shown in Figure 1 (iii). Finally, the junction tree, shown in Figure 1 (iv), is constructed by identifying all the (maximal) cliques in the triangulated graph in Figure 1 (iii) and properly arranging them to satisfy the condition of junction tree. A more formal treatment on triangulation and building junction trees can be found in (Kjaerulff 1990; Huang & Darwiche 1996).

(2) *Initialization.* Every CPD $p(a_i|pa(a_i))$ of the BN will be assigned to a unique node in the junction tree to form a potential if the context of the node contains $\{a_i\} \cup pa(a_i)$. If no CPDs can be assigned to a node h_i in the junction tree, we form a unity potential for h_i defined as $\phi(h_i) = 1$. In our example, the following potentials will be constructed with respect to the junction tree in Figure 1 (iv): $\phi(ac) = p(a) \cdot p(c|a)$, $\phi(bde) = p(b) \cdot p(d|e) \cdot p(e|b)$, $\phi(cdf) = p(f|cd)$, $\phi(def) = 1$, $\phi(fh) = p(h|f)$, $\phi(efg) = p(g|fe)$. In the meantime, a unity potential is formed as well for each separator of the junction tree in Figure 1 (iv) as follows: $\phi^S(c) = 1$, $\phi^S(de) = 1$, $\phi^S(df) = 1$, $\phi^S(ef) = 1$, $\phi^S(f) = 1$. The superscripts S indicate that these potentials are for the separators of junction tree, differentiating them from those potentials for the nodes in the junction tree.

(3) *Local propagation.* The local propagation is a computational scheme based on a primitive operation called *absorption* (or *message passing*) which we review below.

Definition 1 Consider two adjacent nodes R_i, R_j, and their intersection S_{ij} in a junction tree with their respective potentials $\phi(R_i)$, $\phi(R_j)$, and $\phi^S(S_{ij})$. That R_i absorbs R_j (or a message passes from R_j to R_i) means performing the following: (a) Updating $\phi^S(S_{ij})$ by setting $\phi^S(S_{ij}) = \frac{\sum_{R_j - S_{ij}} \phi(R_j)}{\phi^S(S_{ij})}$. (b) Updating $\phi(R_i)$ by setting $\phi(R_i) = \phi(R_i) \cdot \phi^S(S_{ij})$.

The local propagation method is actually a coordinated sequence of absorptions. It begins by picking any node in the junction tree as root, and then perform a sequence of absorptions divided into two passes, namely, the *Collect-Evidence* pass, and the *Distribute-Evidence* pass. The overall control structure for the local propagation method is as follows (Huang & Darwiche 1996):

PROCEDURE Local-Propagation
{
1: Choose a node h_i in the junction tree as root.
2: Unmark all nodes in the junction tree.
Call *Collect-Evidence*(h_i).
3: Unmark all nodes in the junction tree.
Call *Distribute-Evidence*(h_i).
}

During the *Collect-Evidence* pass, each node in the junction tree passes a message to its neighbor towards the root's direction, beginning with the node farthest from the root. During the *Distribute-Evidence* pass, each node in the junction tree passes a message to its neighbor away from the root's direction, beginning with the root itself. The *Collect-Evidence* pass causes $n-1$ messages to be passed. Similarly, the *Distribute-Evidence* pass causes another $n-1$ messages to be passed. Altogether, there are exact $2(n-1)$ messages to be passed (Huang & Darwiche 1996; Jensen 1996).

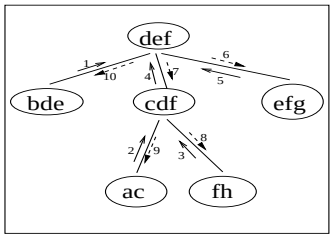

Figure 2: The local propagation procedure applies to the junction tree in Figure 1 (iv) with def as root.

Consider the junction tree for the Asia BN in Figure 1 (iv). Assume the local propagation procedure is applied to the junction tree with the node def picked as root. The execution details of the local propagation procedure can be visualized and depicted in Figure 2. We deliberately draw the junction tree as a rooted tree with root def on top. For simplicity and clarity, we omit the drawing of the separators between adjacent nodes in the junction tree. The local propagation procedure is initiated at the root def with the calling of *Collect-Evidence*, the solid small arrows with numbers attached indicate the sequence of absorptions occurring during the *Collect-Evidence* pass; the dashed small arrows

with number attached indicate the sequence of absorptions occurring during the *Distribute-Evidence* pass. Each arrow, regardless of solid or dashed, indicates an absorption (or message passing) taking place during the execution of local propagation.

After the local propagation procedure finishes its execution, the initial potential attached to each node and separator of the junction tree is now transformed into a marginal distribution. In other words, we obtain the marginals $p(ac)$, $p(cdf)$, $p(bde)$, $p(def)$, $p(fh)$, $p(efg)$ for each node in the junction tree, and the marginals $p(c)$, $p(df)$, $p(de)$, $p(f)$, $p(ef)$ for each separator of the junction tree. The junction tree is now considered to be in a *consistent* state (Jensen 1996).

3. Semi-Join Programs in Relational Databases

In this section, we review the notion of semi-join programs in relational database theory. We begin the discussion by introducing some pertinent notions. Readers are referred to (Maier 1983) for more details of relational database theory.

We define a *database scheme* to be a set $D = \{R_1, \ldots, R_n\}$ of sets of attributes where $\bigcup_{i=1}^{n} R_i = R$. Each R_i is called a *relation scheme*. If $r_1, \ldots, r_n$ are relations over the relation schemes $R_1, \ldots, R_n$, respectively, we then call $d = \{r_1, \ldots, r_n\}$ a *database* over D. We sometimes write $r_i[R_i]$ to explicitly indicate that the relation r_i is over the scheme R_i. By R_iR_j, we mean $R_i \cup R_j$. The relational operators π and $\bowtie$ are used in this paper in their usual sense as *project* and *natural join* operators.

A hypergraph is a pair $(\mathbf{N}, \mathcal{H})$, where $\mathbf{N}$ is a finite set of nodes (attributes) and $\mathcal{H}$ is a set of edges (hyperedges) which are arbitrary subsets of $\mathbf{N}$ (Beeri *et al.* 1983). If the nodes are understood, we will use $\mathcal{H}$ to denote the hypergraph $(\mathbf{N}, \mathcal{H})$. We say an element h_i in a hypergraph $\mathcal{H}$ is a *twig* if there exists another element h_j in $\mathcal{H}$, distinct from h_i, such that $(\cup(\mathcal{H} - \{h_i\})) \cap h_i = h_i \cap h_j$. We call any such h_j a *branch* for the twig h_i. A hypergraph $\mathcal{H}$ is a *hypertree* if its elements can be ordered, say $h_1, h_2, ..., h_n$, so that h_i is a twig in $\{h_1, h_2, ..., h_i\}$, for $i = 2, ..., n-1$. We call any such ordering a *hypertree (tree) construction ordering* for $\mathcal{H}$. It is noted for a given hypertree, there may exist multiple tree construction orderings. Given a tree construction ordering $h_1, h_2, ..., h_n$, we can choose, for each i from 2 to N, an integer $j(i)$ such that $1 \leq j(i) \leq i-1$ and $h_{j(i)}$ is a branch for h_i in $\{h_1, h_2, ..., h_i\}$. We call a function $j(i)$ that satisfies this condition a *branching* for the hypertree $\mathcal{H}$ with $h_1, h_2, ..., h_n$ being the tree construction ordering. For a given tree construction ordering, there might exist multiple choices of branching functions. Given a tree construction ordering $h_1, h_2, ..., h_n$ for a hypertree $\mathcal{H}$ and a branching function $j(i)$ for this ordering, we can construct the following multiset: $\mathcal{L}(\mathcal{H}) = \{h_{j(2)} \cap h_2, h_{j(3)} \cap h_3, ..., h_{j(n)} \cap h_n\}$. The multiset $\mathcal{L}(\mathcal{H})$ is the same for any tree construction ordering and branching function of $\mathcal{H}$ (Beeri *et al.* 1983). We call $\mathcal{L}(\mathcal{H})$ the *separator set* of the hypertree $\mathcal{H}$.

A database scheme D can be naturally viewed as a hypergraph (Beeri *et al.* 1983), each of whose hyperedges is a relation scheme in D. A database scheme D is *acyclic* if its corresponding hypergraph is acyclic. Therefore, we use D to denote both database scheme and its hypergraph representation in the following unless otherwise specified.

1.	$r_4 \leftarrow r_4 \propto r_5$,
2.	$r_2 \leftarrow r_2 \propto r_1$,
3.	$r_6 \leftarrow r_6 \propto r_4$,
4.	$r_5 \leftarrow r_5 \propto r_2$.

Table 1: A SJP.

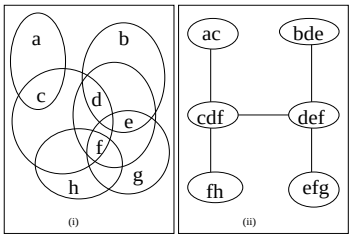

Figure 3: The hypergraph representation of the database scheme D in (i) and one of its junction tree representation in (ii).

Example 1 Consider the database scheme $D = \{R_1 = ac,\ R_2 = cdf,\ R_3 = bde,\ R_4 = def,\ R_5 = fh,\ R_6 = efg\}$. D can be viewed as a hypergraph shown in Figure 3 (i). It can be verified that D is an acyclic database scheme. It has been shown that an acyclic hypergraph may correspond to multiple junction tree [1] representations (Beeri *et al.* 1983), for example, the one in Figure 3 (ii) is a junction tree representation of D with all the separators omitted for clarity.

The following definition of *semi-join* is adapted from (Beeri *et al.* 1983).

Definition 2 Let r_1 and r_2 be two relations over schemes R_1 and R_2, respectively. Consider the following two statements in sequence: (1) $r' = \pi_{R_1 \cap R_2}(r_2)$, (2) $r_1 = r_1 \bowtie r'$. The *semi-join* (SJ) of r_1 and r_2, denoted $r_1 \propto r_2$, is defined as the resulting relation r_1 in statement (2). That is, $r_1 \propto r_2 = r_1 \bowtie \pi_{R_1 \cap R_2}(r_2)$.

Example 2 Consider a database $d = \{r_1, \ldots, r_6\}$ over the database scheme D in Example 1. The SJ of $r_4 \propto r_6$ can be obtained by: (1) compute $r' = \pi_{ef}(r_6)$, where $ef = R_4 \cap R_6$, and (2) compute $r_4 = r_4 \bowtie r'$.

Definition 3 (Maier 1983) Let $d = \{r_1, \ldots, r_n\}$ be a database. A *semi-join program* (SJP) for d is a sequence of assignment statements of the form $r_i \leftarrow r_i \propto r_j$.

Example 3 Consider the database d in Example 2. The sequence of SJs shown in Table 1 is a SJP. By applying the SJP in Table 1 to the database d, updated relations r_4, r_2, r_6 and r_5 will be obtained.

[1] A junction tree is also called a join tree in database theory.

Consider a database $d = \{r_1, \ldots, r_n\}$ over the database scheme $D = \{R_1, \ldots, R_n\}$ such that the natural join of $r_1[R_1], \ldots, r_n[R_n]$ yields $r[R]$, that is:

$$r[R] \quad = \quad r_1[R_1] \bowtie \ldots \bowtie r_n[R_n]. \qquad (1)$$

The database community has dealt with the following problem in the study of answering database queries.

> "Can the projections of $r[R]$ on $R_1, \ldots, R_n$, i.e., $\pi_{R_1}(r), \ldots, \pi_{R_n}(r)$ be computed *without* obtaining $r[R]$ from Eq. (1) in the first place?

The answer to the above problem is *yes* if the database scheme D is acyclic. A special SJP, called a *full reducer* was developed to obtain each $\pi_{R_i}(r)$ from Eq. (1) without obtaining $r[R]$ in advance. Furthermore, $r[R] = \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$.

Definition 4 Let $d = \{r_1, \ldots, r_n\}$ be a database over $D = \{R_1, \ldots, R_n\}$ and SP be a SJP. Let $SP(r_i, d)$ denote the result on r_i after applying SP to d. We call SP a *full reducer* for d if $SP(r_i, d) = \pi_{R_i}(r)$ for each $1 \leq i \leq n$, where $r = r_1 \bowtie \ldots \bowtie r_n$.

It is well known in the database community that a full reducer does *not* always exist for an arbitrary database but it *does* exist for a database if and only if the database scheme is acyclic (Beeri *et al.* 1983). Moreover, the full reducer for an acyclic database can be constructed systematically as described below.

Let $d = \{r_1, \ldots, r_n\}$ be a database over an acyclic database scheme $D = \{R_1, \ldots, R_n\}$. Let JT denote a junction tree for D. We can pick any R_i of D as the root of JT and consider the JT as an oriented tree with root R_i. Perform a *postorder traversal* of JT. Whenever a node R_j of JT is visited, a SJ of the relation $r_k[R_k]$, where R_k is a parent of R_j in JT, with the relation $r_j[R_j]$ for the node R_j, i.e., $r_k \leftarrow r_k \propto r_j$, is added to a SJP denoted SP^+. Let SP^- be the SJP obtained from SP^+ by reversing the order of the steps and changing each $r_i \leftarrow r_i \propto r_j$ to $r_j \leftarrow r_j \propto r_i$. Finally, let SP denote the SJP that equals to SP^+ followed by SP^-. The following Theorem is well established in database theory.

Theorem 1 (Maier 1983) Let $d = \{r_1, \ldots, r_n\}$ be a database over an acyclic database scheme $D = \{R_1, \ldots, R_n\}$. Let JT be a junction tree representation of D. The SJP SP obtained as described above is a full reducer of d. Moreover, SP has $2n - 2$ SJ statements.

	SP^+	SP^-
SP=	1. $r_4 \leftarrow r_4 \propto r_3$, 2. $r_2 \leftarrow r_2 \propto r_1$, 3. $r_2 \leftarrow r_2 \propto r_5$, 4. $r_4 \leftarrow r_4 \propto r_2$, 5. $r_4 \leftarrow r_4 \propto r_6$,	6. $r_6 \leftarrow r_6 \propto r_4$, 7. $r_2 \leftarrow r_2 \propto r_4$, 8. $r_5 \leftarrow r_5 \propto r_2$, 9. $r_1 \leftarrow r_1 \propto r_2$, 10. $r_3 \leftarrow r_3 \propto r_4$.

Table 2: The SJP $SP = SP^+, SP^-$.

Example 4 Consider the database scheme D in Example 1 as shown in Figure 3 and a database d over D. Since D is acyclic, we can construct a semi-join program that is a full reducer. Consider the junction tree in Figure 3 (ii), we pick, say R_4, as the root and redraw the junction tree as an oriented tree rooted at R_4 in Figure 4. The SJP SP^+ is shown in the left column of Table 2, the SJP SP^- is in the right column of Table 2. The complete SJP SP is SP^+ followed by SP^-, i.e. the 10 SJ statements in Table 2. Applying SP to d, we obtain $r_i = \pi_{R_i}(r)$ for $1 \leq i \leq 6$.

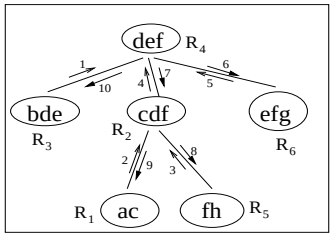

Figure 4: The junction tree with R_4 as root. The numbers on the edge correspond to the statements in the semi-join program SP in Table 2.

One may immediately recognize the similarity between the diagrams in Figure 4 and in Figure 2. This similarity suggests that there exists some intriguing relationship between the semi-join program and the local propagation procedure.

4. Local Propagation is a Generalized Semi-Join Program

In this section, we reveal and prove that local propagation is in fact a generalized semi-join program. We begin the exposition by introducing some notation.

Let x be an instantiation of X. The value of $A \in X$ in x, denoted by $A^{\downarrow x}$, is the *projection* of x onto A. Similarly, we can define the projection of x onto a subset $Z \subseteq X$. An instantiation z is the projection of x onto Z, denoted $z = Z^{\downarrow x}$, if for each $A \in Z$, its value in Z is identical to that of x. Let $\phi(R)$ be a potential and $X \subseteq R$. The *marginal* of $\phi(R)$ on X, denoted $\phi(X)$, is a potential on X such that $\phi(X) = \sum_{R-X} \phi(R)$. Let $\phi(X)$ and $\phi(Y)$ be two potentials. Their *product* is a potential $\phi(W)$, written as $\phi(W) = \phi(X) \cdot \phi(Y)$, where $W = XY$, such that for each value w of W, $\phi(w) = \phi(X^{\downarrow w}) \cdot \phi(Y^{\downarrow w})$. We define the *inverse* function of $\phi(X)$, denoted $\phi^{-1}(X)$ by setting $\phi^{-1}(x) = 1/\phi(x)$ if $\phi(x) \neq 0$, and $\phi^{-1}(x) = \phi(x)$ otherwise, for each value x of X. Obviously, $\phi^{-1}(X)$ is also a potential.

$\mathbf{r}_\Phi(R)$ =

A_1	A_2	.....	A_m	f_{ϕ_R}
t_{11}	t_{12}	.....	t_{1m}	$\phi_R(t_1)$
t_{21}	t_{22}	.....	t_{2m}	$\phi_R(t_2)$
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
t_{s1}	t_{s2}	.....	t_{sm}	$\phi_R(t_s)$

Table 3: The relation $\mathbf{r}_\Phi(R)$ represents a function ϕ_R on $R = \{A_1, A_2, ..., A_m\}$.

Note that each potential $\phi(R)$ over $R = \{A_1, \ldots, A_m\}$ can be represented as a relation (Wong 1997) denoted $\mathbf{r}_\Phi(R)$ as shown in Table 3. Each row in $\mathbf{r}_\Phi(R)$ represents an instantiation of R. The function $\phi(R)$ defines the values of the attribute f_{ϕ_R} in relation $\mathbf{r}_\Phi(R)$. The inverse function of $\phi(R)$, namely, $\phi^{-1}(R)$ can also be represented as a relation denoted $\mathbf{r}_{\Phi^{-1}}(R)$. One may regard that the relation representation of a potential is synonymous with the potential itself. For instance, $\mathbf{r}_\Phi(R)$ is synonymous with the potential $\phi(R)$. We will adopt this relational view of potentials when discussing the local propagation procedure.

We introduce notations for two relational operators corresponding to marginal and product of two potentials (Wong 1997). Let X be a subset of R. The operator of *marginal* is denoted by the symbol Π. The marginal of $\phi(R)$ on X, denoted $\phi(X)$, is itself a potential and can be represented by a relation denoted $\Pi_X(\mathbf{r}_\Phi(R))$, which is synonymous with the marginal $\phi(X)$ (Wong 1997). Consider potentials $\phi(X)$, $\phi(Y)$ and $\phi(W)$ such that $\phi(W) = \phi(X) \cdot \phi(Y)$. Let their respective relation representations be $\mathbf{r}_\Phi(X)$, $\mathbf{r}_\Phi(Y)$, and $\mathbf{r}_\Phi(W)$. The operator of *product join* is denoted by the symbol $\times$. The expression $\phi(W) = \phi(X) \cdot \phi(Y)$ can be conveniently expressed using the product join symbol as $\mathbf{r}_\Phi(W) = \mathbf{r}_\Phi(X) \times \mathbf{r}_\Phi(Y)$. The relation $\mathbf{r}_\Phi(W)$ is synonymous with the potential $\phi(W)$ (Wong 1997).

We now show that the local propagation method is a generalized semi-join program. Consider a BN defined over R with its DAG $\mathcal{D}$. Suppose $\mathcal{D}$ is transformed into a junction tree $JT = \{R_1, \ldots, R_m\}$. Recall the initialization step in the local propagation method, in which a potential is formed for each node and separator of JT. The JPD $p(R)$ can be expressed as follows:

$$p(R) \quad = \quad \phi(R_1) \cdot \phi(R_2) \cdot \ldots \cdot \phi(R_m), \qquad (2)$$

which means $p(R)$ is the product of $\phi(R_i)$. Since $p(R)$, $\phi(R_i)$ in Eq. (2) can be represented as relations $\mathbf{r}_\mathbf{p}(R)$, $\mathbf{r}_\Phi(R_i)$, $i = 1, \ldots, m$, synonymously, we thus can represent Eq. (2) using the product join operator as follows:

$$\mathbf{r}_\mathbf{p}(R) \quad = \quad \mathbf{r}_\Phi(R_1) \times \mathbf{r}_\Phi(R_2) \times \ldots \times \mathbf{r}_\Phi(R_m). \quad (3)$$

The above equation indicates that one can consider $\mathbf{d} = \{\mathbf{r}_\Phi(R_1), \ldots, \mathbf{r}_\Phi(R_m)\}$ as a database over the database scheme $JT = \{R_1, \ldots, R_m\}$. We further call $\mathbf{d}$ a *probabilistic database* of the BN with respect to the junction tree JT.

The Bayesian net community has dealt with the following problem in the study of belief updating.

> Can the marginals of $p(R)$ on R_1, ..., R_n, i.e., $p(R_1)$, ..., $p(R_m)$, be computed *without* obtaining $p(R)$ from Eq. (2) in the first place? Or equivalently, can the marginals of $\mathbf{r}_\mathbf{p}(R)$ on $R_1, \ldots, R_m$, i.e., $\Pi_{R_1}(\mathbf{r}_\mathbf{p}(R))$, ..., $\Pi_{R_m}(\mathbf{r}_\mathbf{p}(R))$ be computed *without* obtaining $\mathbf{r}_\mathbf{p}(R)$ from Eq. (3) in the first place?

The question imposed above is similar to the question imposed in the preceding section for answering database queries which was solved by SJP. The answer to the above problem is also *yes*. The dominant technique used in the Bayesian net community for solving the above problem is the so-called local propagation procedure which seemingly has nothing to do with SJP. However, in the following, we will reveal and prove that the renowned local propagation method is indeed a generalized SJP.

The local propagation procedure is a computational scheme based on the primitive operation *absorption*. We first recall the notion of "absorption" (or *message passing*) in local propagation. Consider two adjacent nodes R_i, R_j, and their intersection S_{ij} in a junction tree with their respective potentials $\phi(R_i)$, $\phi(R_j)$, and $\phi^S(S_{ij})$. Then R_i absorbs R_j by performing the following: (a) updating $\phi^S(S_{ij})$ by setting $\phi^S(S_{ij}) = \frac{\sum_{R_j - S_{ij}} \phi(R_j)}{\phi^S(S_{ij})}$; (b) updating $\phi(R_i)$ by setting $\phi(R_i) = \phi(R_i) \cdot \phi^S(S_{ij})$.

Analogous to the definition of semi-join in databases, we define the notion of *generalized semi-join*.

Definition 5 Let $\mathbf{r}_\phi(R_i)$, $\mathbf{r}_\phi(R_j)$, and $\mathbf{r}_\phi(S_{ij})$ be three relations representing potentials $\phi(R_i)$, $\phi(R_j)$, and $\phi(S_{ij})$, respectively, where $S_{ij} = R_i \cap R_j$. Consider the following two statements in sequence: (1) $\mathbf{r}_\phi(S_{ij}) = \Pi_{S_{ij}}(\mathbf{r}_\phi(R_j)) \times \mathbf{r}_{\phi^{-1}}(S_{ij})$, (2) $\mathbf{r}_\phi(R_i) = \mathbf{r}_\phi(R_i) \times \mathbf{r}_\phi(S_{ij})$. The *generalized semi-join* (GSJ) of $\mathbf{r}_\phi(R_i)$ and $\mathbf{r}_\phi(R_j)$, denoted $\mathbf{r}_\phi(R_i) \propto \mathbf{r}_\phi(R_j)$ is defined as the resulting relation $\mathbf{r}_\phi(R_i)$ in statement (2) above. That is $\mathbf{r}_\phi(R_i) \propto \mathbf{r}_\phi(R_j) = \mathbf{r}_\phi(R_i) \times (\Pi_{S_{ij}}(\mathbf{r}_\phi(R_j)) \times \mathbf{r}_{\phi^{-1}}(S_{ij}))$.

In the above definition, statement (1) corresponds to step (a) of absorption; statement (2) corresponds to step (b) of absorption. Therefore, the notion of GSJ corresponds exactly to the notion of absorption. If in the definition, we initially set the potential $\phi(S_{ij}) = 1$ (thus $\phi^{-1}(S_{ij}) = 1$ as well), then the notions of GSJ and SJ are isomorphic.

Similarly, we define the notion of generalized semi-join program.

Definition 6 Let $\mathbf{d} = \{\mathbf{r}_\Phi(R_1), \ldots, \mathbf{r}_\Phi(R_m)\}$ be a probabilistic database of a BN with respect to a $JT = \{R_1, R_2, \ldots, R_m\}$. A *generalized semi-join program* (GSJP) for $\mathbf{d}$ is a sequence of assignment statements of the form $\mathbf{r}_\Phi(R_i) \leftarrow \mathbf{r}_\Phi(R_i) \propto \mathbf{r}_\Phi(R_j)$ where $1 \leq i \neq j \leq m$.

Definition 7 Let $\mathbf{d} = \{\mathbf{r}_\Phi(R_1), \ldots, \mathbf{r}_\Phi(R_m)\}$ be a probabilistic database of a BN with respect to a junction tree $JT = \{R_1, \ldots, R_m\}$ such that $\mathbf{r}_\mathbf{p}(R) = \mathbf{r}_\Phi(R_1) \times \mathbf{r}_\Phi(R_2) \times \ldots \times \mathbf{r}_\Phi(R_m)$. Let GSP be a GSJP and $GSP(\mathbf{r}_\Phi(R_i), \mathbf{d})$ denote the result on $\mathbf{r}_\Phi(R_i)$ after applying GSP to $\mathbf{d}$. We call a GSP a *generalized full reducer* for $\mathbf{d}$ if $GSP(\mathbf{r}_\Phi(R_i), \mathbf{d}) = \mathbf{r}_\mathbf{p}(R_i)$ for each $1 \leq i \leq n$.

Theorem 2 The local propagation procedure is a generalized semi-join program GSP. Moreover, the GSP is a generalized full reducer which has $2n - 2$ GSJ statements.

Proof: Let $JT = \{R_1, \ldots, R_n\}$ denote a junction tree transformed from a BN. Let $\mathbf{d} = \{\mathbf{r}_\Phi(R_1), \ldots, \mathbf{r}_\Phi(R_m)\}$ be a probabilistic database with respect to JT. The local propagation procedure consists of two rounds of coordinated absorptions (message passings) called *Collect-Evidence* and *Distribute-Evidence* (Jensen 1996). The procedure begins by picking any node, say R_i in JT as a root, calling the routine $Collect - Evidence$ with the root R_i as input, followed

by calling the routine $Distribute-Evidence$ with the root R_i as input argument as well.

During the execution of $Collect-Evidence(R_i)$, a node R_j absorbs a node R_k only if R_k has absorbed all its descendants, otherwise, the recursive call $Collect-Evidence$ on R_k is invoked until the node R_k has no descendants, at which time the recursive call returns and R_k's parent absorbs R_k. This sequence of absorptions is equivalent to a postorder traversal of the rooted JT (with R_i as root) such that whenever a node R_j of JT is visited in the postorder traversal, a GSJ of $\mathbf{r}_\Phi[R_k] \leftarrow \mathbf{r}_\Phi[R_k] \propto \mathbf{r}_\Phi[R_j]$, where R_k is a parent of R_j in the rooted JT, is executed. We collect all the GSJs in sequence and put them in a GSJP denoted GSP^+. There are altogether $n-1$ absorptions each corresponding to a node except the root R_i.

After the routine $Collect-Evidence$ is finished, or equivalently, after all the GSJ statements in GSP^+ have been executed, the local propagation procedure calls the routine $Distribute-Evidence$ with the root R_i as input. The children of R_i in the rooted junction tree absorb R_i respectively. Then the recursive call $Distribute-Evidence$ is invoked on each of R_i's child until this recursive calling reaches the leaf node in the rooted junction tree. This sequence of absorptions is equivalent to a set GSP^- of GSJ statements obtained from GSP^+ by reversing the order of the steps and changing each $\mathbf{r}_\Phi[R_i] \leftarrow \mathbf{r}_\Phi[R_i] \propto \mathbf{r}_\Phi[R_j]$ to $\mathbf{r}_\Phi[R_j] \leftarrow \mathbf{r}_\Phi[R_j] \propto \mathbf{r}_\Phi[R_i]$. There are altogether $n-1$ absorptions in GSP^-.

Finally, let GSP denote the GSJP that equals to GSP^+ followed by GSP^-. The GSP contains $2n-2$ GSJ statements each corresponding to an absorption in the exact order that the absorption occurred during the two rounds of message passings in local propagation. In other words, the local propagation procedure is actually the GSP. The GSP is a generalized full reducer of $\mathbf{d}$ since the local propagation produces $p(R_i)$ after its two rounds of message passings.

5. Conclusion

Without realizing the connection revealed in this paper, in the late 1980s and the early 1990s, researchers in the Bayesian net community independently developed and matured the local propagation technique based on which belief updating in BNs are conducted (Huang & Darwiche 1996). However, our study shows that the local propagation method used in Bayesian nets is in fact a generalized semi-join program in relational databases. This connection has several implications to the current conduct of belief updating in BNs. (1) The development of semi-join program in relational databases was originally for the purpose of answering database queries. Since local propagation can be considered as a (generalized) semi-join program, this connection implies that we may reconsider the task of belief updating, i.e., computing the posterior probabilities, from a database perspective. Some initial effort has been reported. In (Wong, Wu, & Butz 2003), a method which was originally used in relational database for answering database queries has been successfully adapted for belief updating in Bayesian nets so that repetitive applications of the local propagation procedure can be avoided. (2) Various efforts have been spent on improving the efficiency of using local propagation for belief updating (Kjaerulff 1990; Jensen & Jensen 1994; Kjaerulff 1997; Madsen & Jensen 1998; Olesen & Madsen 2002). Because of the connection between local propagation and semi-join program, one may look into the literature of database theory and take advantages of query optimization techniques used in relational database theory to further improve the efficiency of belief updating in BNs. (3) The fact that the local propagation is indeed a semi-join program further confirms the intriguing relationship between BNs and relational databases.

References

Beeri, C.; Fagin, R.; Maier, D.; and Yannakakis, M. 1983. On the desirability of acyclic database schemes. *Journal of the ACM* 30(3):479–513.

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

Jensen, F. V., and Jensen, F. 1994. Optimal junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, 360–366.

Jensen, F. 1996. *An Introduction to Bayesian Networks*. UCL Press.

Kjaerulff, U. 1990. Triangulation of graphs—algorithms giving small total state space. Technical report, JUDEX, Aalborg, Denmark.

Kjaerulff, U. 1997. Nested junction trees. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, 302–313. Morgan Kaufmann Publishers.

Lauritzen, S., and Spiegelhalter, D. 1988. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society* 50:157–244.

Madsen, A. L., and Jensen, F. V. 1998. Lazy propagation in junction trees. In Cooper, G. F., and Moral, S., eds., *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 362–369. San Francisco: Morgan Kaufmann.

Maier, D. 1983. *The Theory of Relational Databases*. Principles of Computer Science. Rockville, Maryland: Computer Science Press.

Olesen, K., and Madsen, A. 2002. Maximal prime subgraph decomposition of bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics* 32(1):21–31.

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

Wong, S.; Wu, D.; and Butz, C. 2003. Probabilistic reasoning in bayesian networks: a relational database perspective. In *The 16th Canadian Conference on Artificial Intelligence Halifax, Canada, to appear*.

Wong, S. 1997. An extended relational data model for probabilistic reasoning. *Journal of Intelligent Information Systems* 9:181–202.