Join Tree Propagation with Prioritized Messages

C. J. Butz, S. Hua, K. Konkel, and H. Yao
Department of Computer Science, University of Regina, Regina, Canada S4S 0A2

Current join tree propagation algorithms treat all propagated messages as being of equal importance. On the contrary, it is often the case in real-world Bayesian networks that only some of the messages propagated from one join tree node to another are relevant to subsequent message construction at the receiving node. In this article, we propose the first join tree propagation algorithm that identifies and constructs the relevant messages first. Our approach assigns lower priority to the irrelevant messages as they only need to be constructed so that posterior probabilities can be computed when propagation terminates. Experimental results, involving the processing of evidence in four real-world Bayesian networks, empirically demonstrate an improvement over the state-of-the-art method for exact inference in discrete Bayesian networks. © 2009 Wiley Periodicals, Inc. NETWORKS, Vol. 55(4), 350–359 2010

Keywords: Bayesian networks; join tree propagation; exact probabilistic inference; conditional independence

1. INTRODUCTION

Bayesian networks [19] are a formal approach to uncertainty management based on the combination of probability theory and graph theory. A Bayesian network is a directed acyclic graph [26] coupled with a set of conditional probability tables [23]. The vertices of the graph represent random variables, while the arcs in the graph represent probabilistic dependencies amongst the variables. It can be shown [11] that the product of the conditional probability tables is a joint probability distribution by utilizing the probabilistic conditional independencies [25] encoded in the directed acyclic graph. By providing a sound and concise representation of probabilistic knowledge [7, 13], Bayesian networks have been successfully applied in practice to many problem domains, including medical diagnosis [12, 20].

Although Cooper [9] has shown that the complexity of inference in Bayesian networks is NP-hard, various approaches have been developed that seem to work quite well in practice [5, 8, 10, 22, 24]. Generally speaking, there are two approaches to exact inference in discrete Bayesian networks. One approach, called direct computation, performs inference directly within a Bayesian network. Two classical direct computation algorithms are variable elimination (VE) [28] and arc reversal (AR) [18, 21]. Another approach, known as join tree propagation, first builds a secondary network called a join tree [4] by triangulating [27] the directed acyclic graph. Join tree propagation then performs inference by propagating probabilities in the join tree. Shafer [23] emphasizes that join tree probability propagation is central to the theory and practice of probabilistic expert systems.

The state-of-the-art algorithm for exact inference in discrete Bayesian networks is Madsen's Lazy AR [14, 15]. Lazy AR is a hybrid approach, because it falls into the join tree propagation framework while utilizing a direct computation inference algorithm. More specifically, AR is applied to physically construct the messages passed from a join tree node to a neighbor. Although traditional join tree propagation algorithms pass a single probability table between neighboring nodes in the join tree [23], Lazy AR can pass a set of probability tables between nodes. By maintaining a factorization of probability tables, Lazy AR can show favorable experimental results [14, 15] with its exploitation of barren [21] variables and independencies induced by evidence. Nevertheless, there is room for improvement, because Lazy AR views all propagated messages as being of equal importance. On the contrary, it is often the case in real-world Bayesian networks that only some of the messages propagated from one join tree node to a neighbor are relevant to subsequent message construction at the neighbor node.

In this article, we propose prioritized join tree propagation as a new approach to exact inference in Bayesian networks. The distinguishing feature of our method can be seen as a three-step process. First, AR is applied only to determine the distribution heading (the schema or label) of each message (distribution) that will be propagated in a join tree. In other words, we identify all of the message headings without physically building in computer memory the probability distributions complete with probability values. Second, the relevant and irrelevant messages are identified (with respect to the subsequent message construction at the receiving join tree nodes). Finally, the relevant messages are then physically constructed in memory using variable elimination. Thus, in

Received September 2006; accepted February 2009
Correspondence to: C. J. Butz; e-mail: butz@cs.uregina.ca
Contract grant sponsor: NSERC Discovery; Contract grant number: 238880
DOI 10.1002/net.20328
Published online 31 July 2009 in Wiley InterScience (www.interscience.wiley.com).

a	$p(a)$
1	0.496

c	$p(c)$
1	0.577

d	e	f	$p(f\|d,e)$
0	0	1	0.710
0	1	1	0.193
1	0	1	0.485
1	1	1	0.602

b	f	g	$p(g\|b,f)$
0	0	1	0.027
0	1	1	0.123
1	0	1	0.898
1	1	1	0.405

g	h	i	j	$p(j\|g,h,i)$
0	0	0	1	0.178
0	0	1	1	0.565
0	1	0	1	0.446
0	1	1	1	0.729
1	0	0	1	0.931
1	0	1	1	0.582
1	1	0	1	0.403
1	1	1	1	0.222

a	b	$p(b\|a)$
0	1	0.052
1	1	0.358

c	d	$p(d\|c)$
0	1	0.714
1	1	0.627

c	e	$p(e\|c)$
0	1	0.383
1	1	0.286

c	h	$p(h\|c)$
0	1	0.214
1	1	0.651

h	i	$p(i\|h)$
0	1	0.104
1	1	0.369

g	k	$p(k\|g)$
0	1	0.593
1	1	0.416

FIG. 1. Tables $p(a)$, $p(b|a)$, $p(c)$, $p(d|c)$, $p(e|c)$, $p(f|d,e)$, $p(g|b,f)$, $p(h|c)$, $p(i|h)$, $p(j|g,h,i)$, and $p(k|g)$.

our method, a join tree node is allowed to physically build an outgoing message as soon as it has received all incoming messages that are relevant to its construction. The main advantage of prioritized join tree propagation over Lazy AR is that our approach does not force a join tree node to wait for messages that are irrelevant to its subsequent message computation. The efficiency improvement offered by prioritized join tree propagation is shown through empirical evaluations on four real-world Bayesian networks and one benchmark Bayesian network. As is usually done, inference is performed in each network with varying amounts of evidence, namely, zero, nine, and eighteen percent. Our prioritized join tree propagation approach finished inference faster than Lazy AR in all fifteen cases, as reported in Tables 2, 3, and 4.

This article is organized as follows. Section 2 contains background information. We propose prioritized join tree propagation in Section 3. Experimental results are shown in Section 4. Section 5 presents the conclusion.

2. DEFINITIONS

Here we review results from discrete Bayesian networks, and three approaches for exact inference therein.

Consider a finite set of discrete random variables $U = \{v_1, v_2, \dots, v_n\}$. Let $\mathrm{dom}(v_i)$ denote the finite domain of values that each variable $v_i \in U$ can assume. For a subset $X \subseteq U$, the Cartesian product of the domains of the individual variables in X is $\mathrm{dom}(X)$. An element $x \in \mathrm{dom}(X)$ is a configuration of X. A potential [12] on $\mathrm{dom}(X)$ is a function ϕ such that $\phi(x) \geq 0$, for each configuration $x \in \mathrm{dom}(X)$, and at least one $\phi(x)$ is positive. For simplicity, we speak of a potential as defined on X instead of on $\mathrm{dom}(X)$, and we call X its domain rather than $\mathrm{dom}(X)$ [23]. A joint probability distribution [23] on U, written $p(U)$, is a function p on U satisfying the following two conditions: (i) $0 \leq p(u) \leq 1$, for each configuration $u \in \mathrm{dom}(U)$; (ii) $\sum_{u \in \mathrm{dom}(U)} p(u) = 1$. Let X and Y be two disjoint subsets of U. A conditional probability table (table for short) [23] for Y given X, denoted $p(Y|X)$, is a nonnegative function on $X \cup Y$, satisfying the following condition: for each configuration $x \in \mathrm{dom}(X)$, $\sum_{y \in \mathrm{dom}(Y)} p(Y = y|X = x) = 1$. For example, given binary variables $U = \{a, b, \dots, k\}$, tables $p(a)$, $p(b|a)$, $p(c)$, $p(d|c)$, $p(e|c)$, $p(f|d,e)$, $p(g|b,f)$, $p(h|c)$, $p(i|h)$, $p(j|g,h,i)$, and $p(k|g)$ are shown in Figure 1. Note that missing probabilities can be obtained by the definition of a table. For instance, $p(a = 0) = 0.504$ and $p(b = 0|a = 0) = 0.948$.

The heading of a table is the label shown above the probability column. For instance, the heading of table $p(a)$ in Figure 1 is the label "$p(a)$" appearing above the probability column. It will always be made clear as to whether $p(Y|X)$ refers to the heading or the table itself.

A discrete Bayesian network [19] on $U = \{v_1, v_2, \dots, v_n\}$ is a pair (D, C). D is a directed acyclic graph with vertex set U. C is the set of tables $\{p(v_i|P_i)|i = 1, 2, \dots, n\}$, where P_i denotes the parents of variable $v_i \in D$. For example, the directed acyclic graph in Figure 2i together with the corresponding tables in Figure 1 is based on a real-world Bayesian network for coronary heart disease (CHD) [12]. Here, the parents P_i of variable $v_i = g$ are $P_i = \{b, f\}$.

We use the terms Bayesian network and directed acyclic graph interchangeably. A topological ordering [7] of the variables in a Bayesian network is denoted by $\prec$. The family F_i of a variable v_i in a Bayesian network is the variable together with its parents, that is, $\{v_i\} \cup P_i$. A variable without parents is called a root variable.

A Bayesian network D graphically encodes probabilistic conditional independencies [25], which can be inferred from D using the d-separation algorithm [19]. Based on the independencies encoded in D, the product of the tables in C is a joint distribution $p(U)$ [11], namely, $p(U) = \prod_{v_i \in U} p(v_i|P_i)$. For example, the independencies encoded in the Bayesian network of Figure 2i indicate that the product of the tables in Figure 1 is a joint distribution on $U = \{a, b, c, d, \dots, k\}$, namely, $p(U) = p(a) \cdot p(b|a) \cdot p(c) \cdot p(d|c) \cdots p(k|g)$. Thereby, one favorable feature of Bayesian networks is that they provide a compact, graphical representation of a joint distribution modelling a real-world problem domain. For instance, only 30 probabilities are required for the CHD Bayesian network in Figure 2i versus $2^{11} - 1$ probabilities required for specifying the joint distribution $p(U)$ directly.

Suppose that the values e of a set E of variables in a Bayesian network have been observed and that the posterior probabilities of set X (disjoint with E) are sought. All variables outside of $E \cup X$ must necessarily be eliminated in answering this query, denoted $p(X|E = e)$. A brute-force approach to eliminating these variables, however, can involve unnecessary manipulation of probability distributions in memory. Given a Bayesian network D and a query

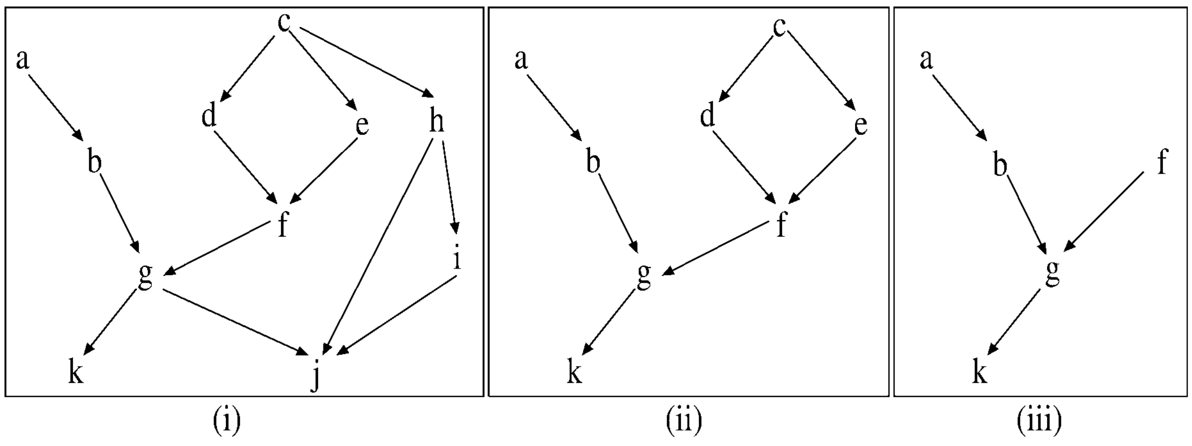


FIG. 2. (i) The coronary heart disease (CHD) Bayesian network [12]. Given the query $p(k|f = 0)$: (ii) barren variables h, i, and j have been pruned; (iii) variables c, d, and e are also removed as they are independent of k given evidence $f = 0$.

$p(X|E=e)$, a variable v_i is barren [21] if $(\{v_i\} \cup Y) \cap (X \cup E) = \emptyset$, where Y is the set of descendants of v_i in D. For example, given the query $p(k|f = 0)$ posed to the Bayesian network in Figure 2i, variables h, i, and j are barren. Thus, they can be removed, yielding the network in Figure 2ii.

Similarly, independencies induced by evidence can also be taken advantage of to save unnecessary physical computation. Baker and Boult [2] proposed an algorithm, which we will call Prune, that prunes all variables from a Bayesian network that are irrelevant to a given query $p(X|E = e)$. Their algorithm removes barren variables as well as those variables rendered immaterial to X given the evidence $E = e$. Note that the time complexity of Prune is $O(|\lambda|)$, where $|\lambda|$ is the number of arcs in the Bayesian network [2]. For example, given evidence $f = 0$ in query $p(k|f = 0)$ posed to Figure 2ii, variable k is conditionally independent of variables c, d, and e. Thus, c, d, and e can be safely removed to yield the smaller Bayesian network in Figure 2iii.

A root variable v_i that is also an evidence variable can have its table ignored and, for each child v_j of v_i, the table $p(v_j|P_j)$ is modified to agree with the observed evidence. In our running example, because f is both an evidence variable and a root variable in Figure 2iii, table $p(f|d, e)$ is ignored and table $p(g|b,f)$ for the child g of f is modified to only contain rows agreeing with the evidence $f = 0$ [23]. That is, all rows in $p(g|b,f)$ with $f = 1$ are deleted leaving $p(g|b,f = 0)$ stored in computer memory. The query $p(k|f = 0)$ can now be answered by eliminating variables a, b, and g from the distributions stored in computer memory.

Given a set of variables to be eliminated from a set S of potentials, variable elimination (VE) [28] recursively eliminates the variables v_i one-by-one using the following four steps: (i) remove from S the set of potentials containing v_i; (ii) multiply together the distributions removed from S; (iii) sum v_i out of the potential obtained in (ii); and (iv) add the resulting potential to S. Unlike VE, arc reversal (AR) [18, 21] eliminates variables while maintaining a factorization of tables. The following outline draws from [14, 15, 21]. Suppose variable v_i is to be eliminated. Arc (v_i, v_j) is reversed by setting the new parents of v_j as $P_i \cup P_j - \{v_i\}$, while making $P_i \cup F_j - \{v_i\}$ the new parents of v_i. Next, new tables for v_j and v_i in the modified directed acyclic graph are physically constructed as follows [21]:

$$p(v_j|P_i \cup P_j - \{v_i\}) = \sum_{v_i} p(v_i|P_i) \cdot p(v_j|P_j), \qquad (1)$$

and

$$p(v_i|P_i \cup F_j - \{v_i\}) = \frac{p(v_i|P_i) \cdot p(v_j|P_j)}{p(v_j|P_i \cup P_j - \{v_i\})}. \qquad (2)$$

Note that it is not necessary to evaluate Equation (2) when the final arc from v_i is reversed, because v_i will be eliminated. Also, a directed acyclic graph structure is maintained after eliminating a variable, because AR uses a fixed topological ordering of the original Bayesian network to avoid creating directed cycles.

Although VE and AR compute the posterior probabilities of a set X of variables given the evidence $E = e$, join tree propagation seeks to update all variables in $U - E$. The notion of a join tree is first needed. A join tree [4] is a tree with sets of variables as nodes, and with the property that any variable in two nodes is also in any node on the path between the two. The directed acyclic graph D of a Bayesian network is converted into a join tree via the moralization and triangulation procedures. The moralization [19] D_{m} of D is the undirected graph obtained from D by adding undirected edges between every pair of nonadjacent vertices that have a common child, and then dropping the directions of all arcs. If necessary, undirected edges are added to the moralization to create a triangulated graph. An undirected graph is triangulated (chordal) [27], if each cycle of length four or more possesses an edge (v_i, v_j) between two nonadjacent variables v_i and v_j in the cycle. Finding a triangulation of a graph by adding the minimum number of edges is NP-complete [19]. Each maximal clique (complete subgraph) [6] of the triangulated graph is represented by a node in the join tree. Although the CHD Bayesian network is useful for illustrating some pertinent concepts, it is not interesting due to its small size. A

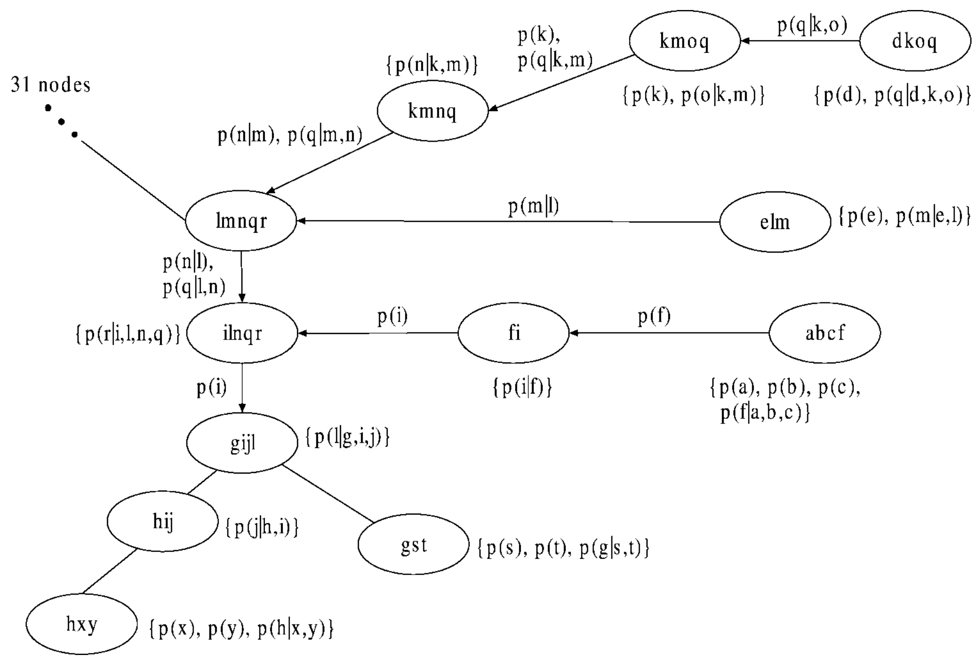


FIG. 3. The Hailfinder join tree, where only the pertinent nodes and messages are shown. Both messages $p(n|l)$ and $p(q|l,n)$ from *lmnqr* are irrelevant to the forwarding of message $p(i)$ at node *ilnqr*.

larger real-world Bayesian network, called Hailfinder [1], is instead used here. Figure 3 shows the partial depiction of one possible join tree for Hailfinder. Some join tree edges have been directed to depict the propagation of those messages pertinent to our forthcoming discussion. Each join tree node name corresponds to the variables in the node. For instance, in Figure 3, node *abcf* means that the join tree node consists of variables $\{a, b, c, f\}$. The table of each variable v_i in the given Bayesian network is assigned to precisely one join tree node containing v_i and its parents P_i. For instance, $p(f|a, b, c)$ is assigned to *abcf* in Figure 3.

Given collected evidence, messages are systematically passed in a join tree such that each join tree node can compute the posterior probabilities of its variables when propagation finishes. In particular, the message passing is controlled by the rule that each node waits to send its message to a particular neighbor until it has received all messages from all its other neighbors. It is well known that join tree propagation can be performed serially or in parallel [16, 23]. Our discussion is based on parallel computation.

Madsen's Lazy AR [14, 15] implements AR as the engine for performing inference in join tree propagation. When a join tree node N is ready to send its messages to a particular neighbor N', the Lazy AR approach computes the messages from node N to N' using the following three steps: (i) collect all messages from N's other neighbors; (ii) identify the relevant and irrelevant variables; (iii) apply AR to physically eliminate variables in $N - N'$ from the relevant distributions.

Example 1. *Consider how Lazy AR physically constructs the messages passed from node lmnqr to node ilnqr in the Hailfinder join tree of Figure 3 given evidence* $j = 0$. *In Step (i), lmnqr collects* $p(n|m)$ *and* $p(q|m, n)$ *from node kmnq, as well as* $p(m|l)$ *from node elm. For Step (ii), no variables are*

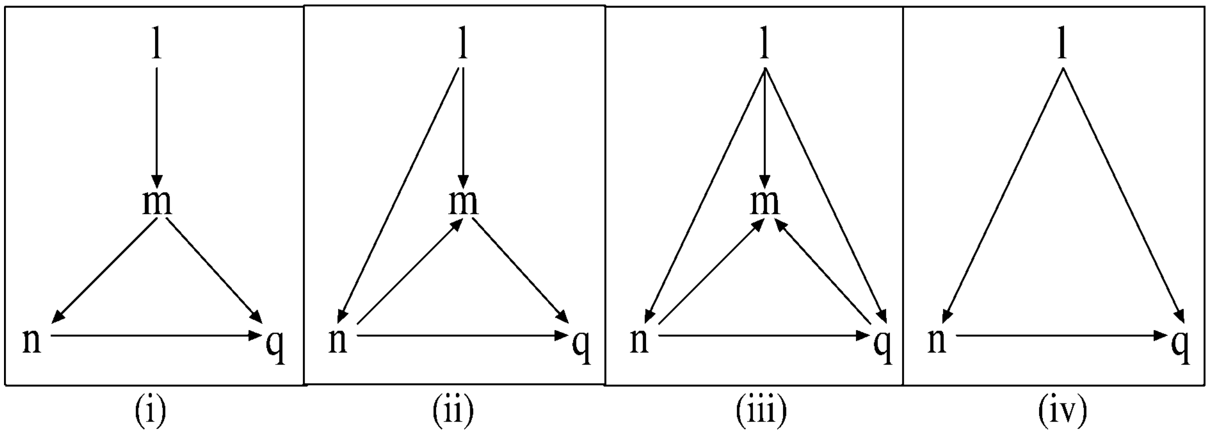


FIG. 4. Lazy AR applies AR to eliminate variable m in Example 1.

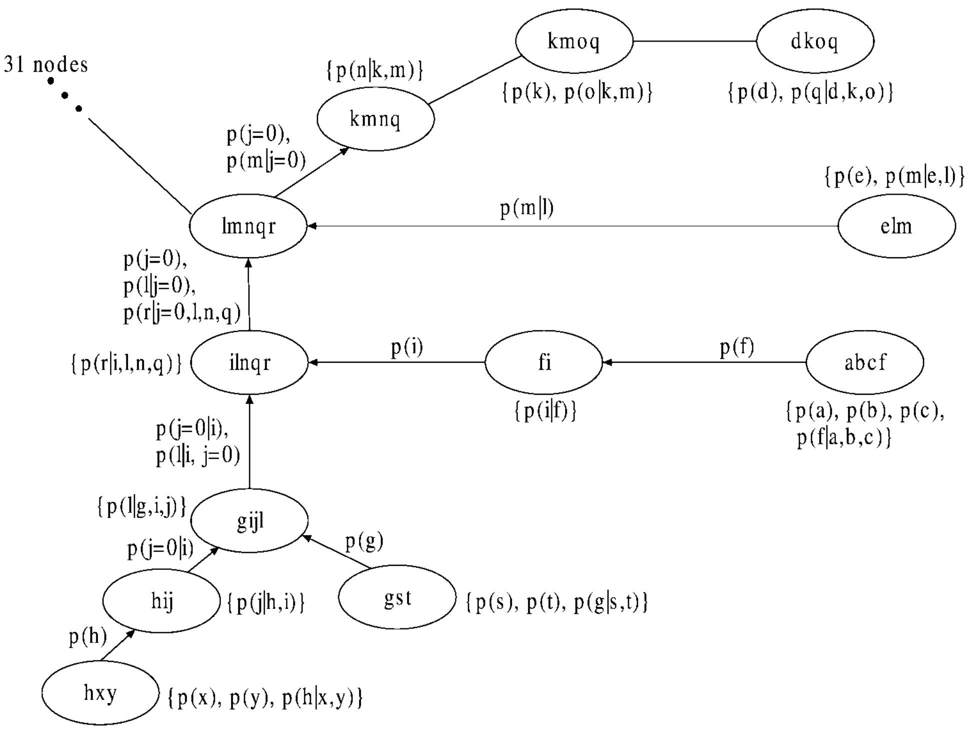


FIG. 5. For building messages $p(j = 0)$ and $p(m|j = 0)$ from node *lmnqr*, messages $p(j = 0)$ and $p(l|j = 0)$ from *ilnqr* are, respectively, relevant, whereas the message $p(r|j = 0, l, n, q)$ is irrelevant in both cases.

irrelevant. In Step (iii), Lazy AR eliminates variable m from the Bayesian network in Figure 4i defined by these tables as follows. Observe that arcs (m, n) and (m, q) need to be reversed and $n \prec q$. For the first arc (m, n), $v_i = m$, $P_i = \{l\}$, $v_j = n$, $P_j = \{m\}$, and $F_j = \{m, n\}$. The reversed arc (n, m) is created by setting $P_i = \{l, n\}$ and $P_j = \{l\}$, as shown in Figure 4ii. Next, the new table $p(n|l)$ for n is physically constructed by Equation (1) as: $p(n|l) = \sum_m p(m|l) \cdot p(n|m)$. Using Equation (2), the new table for m is constructed in computer memory as: $p(m|l, n) = (p(m|l) \cdot p(n|m))/p(n|l)$. Similarly, arc (m, q) is reversed as (q, m) and a new table for q is physically computed: $p(q|l, n) = \sum_m p(m|l, n) \cdot p(q|m, n)$, as shown in Figure 4iii. Variable m can now be removed from the network as illustrated in Figure 4iv. The constructed messages $p(n|l)$ and $p(q|l, n)$ are sent to ilnqr.

3. PRIORITIZED JOIN TREE PROPAGATION

In this section, we introduce prioritized join tree propagation as a new approach to Bayesian inference. Our method can be broken down into three tasks: (i) identify the headings of all messages to be propagated in the join tree. (ii) for each join tree node, identify the relevant and irrelevant incoming messages with respect to each outgoing message; (iii) apply VE to physically build the relevant messages, followed by the irrelevant messages.

For probabilistic inference in real-world Bayesian networks, it is often the case that only some of the messages propagated are relevant to subsequent message computation at the receiving node. Using the propagation of evidence in the Hailfinder join tree, Example 2 will show that all messages from *lmnqr* are irrelevant to *ilnqr*, while Example 3 illustrates that some of the messages from *ilnqr* to *lmnqr* are irrelevant. For simplicity we ignore all distributions not essential to our main point.

Example 2. *Consider the message $p(i)$ to be passed from node ilnqr to node gijl in Figure 3. Clearly, ilnqr simply forwards to gijl the incoming message $p(i)$ from node fi. Therefore, all of the physical computation done by Lazy AR at node lmnqr in Example 1 to construct messages $p(n|l)$ and $p(q|l, n)$ is irrelevant to the subsequent message construction at ilnqr.*

Example 2 shows that Lazy AR will force node ilnqr to wait for node *lmnqr* to build messages $p(n|l)$ and $p(q|l, n)$, even though these messages are irrelevant to the forwarding of the subsequent message $p(i)$ from *ilnqr*.

Example 3. *In Figure 5, consider Lazy AR's construction of the three messages $p(j = 0)$, $p(l|j = 0)$, and $p(r|j = 0, l, n, q)$ sent from node ilnqr to node lmnqr in the Hailfinder join tree given evidence $j = 0$. By Step (i), node ilnqr has collected messages $p(i)$, $p(j = 0|i)$, and $p(l|i, j = 0)$ from its other neighbors. In Step (ii), all variables are relevant. For Step (iii), variable i needs to be eliminated from these three messages together with the table $p(r|i, l, n, q)$ assigned to ilnqr. Arcs (i,j), (i, l), and (i, r) need to be reversed. Because $j \prec l \prec r$, AR is applied as follows. Arc (i,j) is reversed as (j, i) using $p(j = 0) = \sum_i p(i) \cdot p(j = 0|i)$ and $p(i|j = 0) = (p(i) \cdot p(j = 0|i))/p(j = 0)$. To create (l, i),* *Lazy AR builds $p(l|j = 0) = \sum_i p(i|j = 0) \cdot p(l|i,j = 0)$ and $p(i|j = 0, l) = (p(i|j = 0) \cdot p(l|i,j = 0))/p(l|j = 0)$. Lastly, reversing (i, r) is accomplished by physically building $p(r|j = 0, l, n, q) = \sum_i p(i|j = 0, l) \cdot p(r|i, l, n, q)$. Lazy AR then sends the constructed messages $p(j = 0)$, $p(l|j = 0)$, and $p(r|j = 0, l, n, q)$ to lmnqr. Revisiting Figure 5, now consider how the subsequent messages $p(j = 0)$ and $p(m|j = 0)$ from lmnqr to node kmnq are constructed. Message $p(j = 0)$ can be simply forwarded meaning that the incoming messages $p(l|j = 0)$, $p(r|j = 0, l, n, q)$, and $p(m|l)$ are irrelevant. Lazy AR builds message $p(m|j = 0)$ as: $p(m|j = 0) = \sum_l p(l|j = 0) \cdot p(m|l)$. This implies that only the incoming message $p(l|j = 0)$ from ilnqr and message $p(m|l)$ from elm are relevant, while the incoming messages $p(j = 0)$ and $p(r|j = 0, l, n, q)$ are irrelevant.*

Example 3 shows that Lazy AR forces *lmnqr* to wait for all messages to be received, even though only some of the messages are relevant to subsequent message computation at *lmnqr*. Examples 2 and 3 together motivate the development of a new approach to Bayesian inference—one that gives priority to relevant messages.

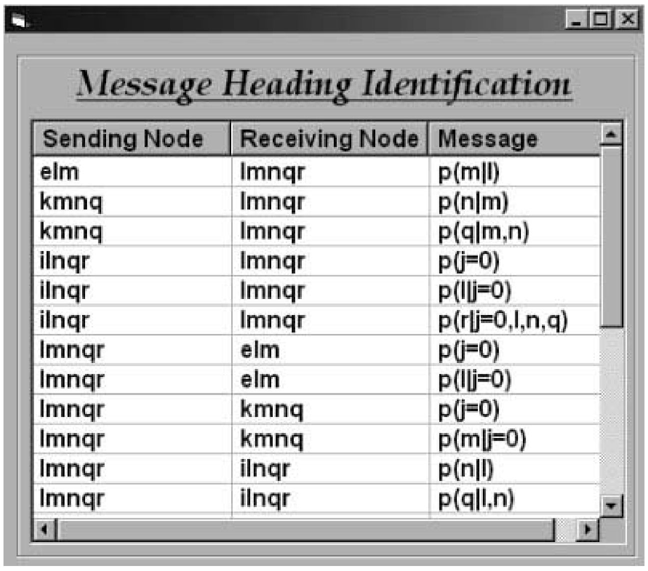
3.1. Identifying the Message Headings

Our prioritized join tree propagation approach can identify the headings of all messages propagated in a join tree with Algorithm 1, called MsgId, defined as follows.

Algorithm 1 [3] MsgId (C, X)

```
Input: $C$- a set of distribution headings at a join tree node $N$,
       $X$- the set of variables to be eliminated.
Output: the set $C$ of message headings sent from a join tree
        node to a neighbor.
begin
Construct the unique directed acyclic graph $D_N$ defined
by $C$.
for each variable $v_i$ in $X$
    Let $Y = \{v_1, \ldots, v_k\}$ be the set of all children of $v_i$ in
    $D_N$, where $v_1 \prec \cdots \prec v_k$.
    for $j = 1, \ldots, k$
        $P_j = P_i \cup P_j - \{v_i\}$.
        $P_i = P_i \cup F_j - \{v_i\}$.
    Remove $v_i$ from $D_N$ and its distribution heading
    from $C$.
return($C$)
end
```

Example 4. *In the Hailfinder join tree of Figure 3, let us show how node lmnqr identifies the headings of the messages to be sent to its neighbor ilnqr. Node lmnqr collects the headings $p(m|l)$, $p(n|m)$, and $p(q|m, n)$ sent from its neighbors elm and kmnq. It then calls MsgId with $C = \{p(m|l), p(n|m), p(q|m, n)\}$ and $X = \{m\}$. Here, $v_i = m$, $Y = \{v_1 = n, v_2 = q\}$, $P_i = \{l\}$, $P_1 = \{m\}$, $F_1 = \{m, n\}$, $P_2 = \{m, n\}$, and $F_2 = \{m, n, q\}$. Consider the first variable $v_1 = n$ in Y. As $P_i \cup P_1 - \{v_i\} = \{l\}$, P_1 is modified as $P_1 = \{l\}$. Because $P_i \cup F_1 - \{v_i\} = \{l, n\}$, P_i is changed to $P_i = \{l, n\}$. The set C is set to $C = \{p(m|l, n), p(n|l), p(q|m, n)\}$. Now consider the second variable $v_2 = q$ in Y. P_2 is changed to $\{l, n\}$, because $P_i \cup P_2 - \{v_i\} = \{l, n\}$. As $P_i \cup F_2 - \{v_i\} = \{l, n, q\}$, P_i is modified as $P_i = \{l, n, q\}$. Variable m is then removed. Thus, $C = \{p(n|l), p(q|l, n)\}$ denotes the headings of messages $p(n|l)$ and $p(q|l, n)$ sent from lmnqr to ilnqr.*



Sending Node	Receiving Node	Message
elm	lmnqr	p(m|l)
kmnq	lmnqr	p(n|m)
kmnq	lmnqr	p(q|m,n)
ilnqr	lmnqr	p(j=0)
ilnqr	lmnqr	p(l|j=0)
ilnqr	lmnqr	p(r|j=0,l,n,q)
lmnqr	elm	p(j=0)
lmnqr	elm	p(l|j=0)
lmnqr	kmnq	p(j=0)
lmnqr	kmnq	p(m|j=0)
lmnqr	ilnqr	p(n|l)
lmnqr	ilnqr	p(q|l,n)

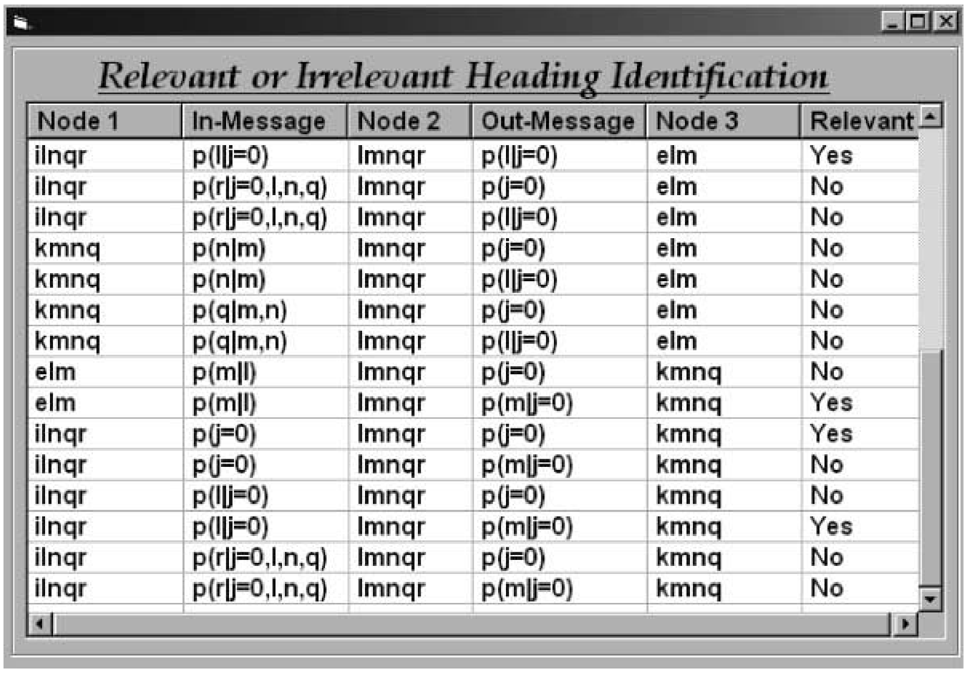
FIG. 6. Given evidence $j = 0$ in the join tree for the real-world Hailfinder Bayesian network (see Figs. 3 and 5), our system identifies the headings of the messages (distributions) to be propagated.

Note that it is easy to use MsgId to identify message headings when evidence is propagated in a join tree. Given evidence $E = e$, set $N = N \cup E$ for each node N in the join tree. On this augmented join tree, apply MsgId as usual. After message heading identification, for each evidence variable $v \in E$, change each occurrence of v in the headings returned by MsgId from v to $v = \varepsilon$, where ε is the observed value of v. The screen-shot of our implemented system in Figure 6 shows some of the message headings identified given evidence $j = 0$ in the Hailfinder join tree. Figure 6 emphasizes node *lmnqr* and its neighbors. The important point is that the MsgId algorithm identifies the headings of the messages that will be propagated in the join tree before the probability distributions of the messages are physically built in computer memory.

3.2. Identifying the Relevant Messages

Here, we determine the relevant messages with respect to subsequent message computation at the receiving node.

Consider three distinct join tree nodes N_1, N_2, and N_3 such that $p(X_1|Y_1)$ is a message to be passed from N_1 to N_2 and $p(X_2|Y_2)$ is a message to be passed from N_2 to N_3. We say $p(X_1|Y_1)$ is relevant to $p(X_2|Y_2)$, if $p(X_1|Y_1)$ is required in



Relevant or Irrelevant Heading Identification

Node 1	In-Message	Node 2	Out-Message	Node 3	Relevant
ilnqr	p(l|j=0)	lmnqr	p(l|j=0)	elm	Yes
ilnqr	p(r|j=0,l,n,q)	lmnqr	p(j=0)	elm	No
ilnqr	p(r|j=0,l,n,q)	lmnqr	p(l|j=0)	elm	No
kmnq	p(n|m)	lmnqr	p(j=0)	elm	No
kmnq	p(n|m)	lmnqr	p(l|j=0)	elm	No
kmnq	p(q|m,n)	lmnqr	p(j=0)	elm	No
kmnq	p(q|m,n)	lmnqr	p(l|j=0)	elm	No
elm	p(m|l)	lmnqr	p(j=0)	kmnq	No
elm	p(m|l)	lmnqr	p(m|j=0)	kmnq	Yes
ilnqr	p(j=0)	lmnqr	p(j=0)	kmnq	Yes
ilnqr	p(j=0)	lmnqr	p(m|j=0)	kmnq	No
ilnqr	p(l|j=0)	lmnqr	p(j=0)	kmnq	No
ilnqr	p(l|j=0)	lmnqr	p(m|j=0)	kmnq	Yes
ilnqr	p(r|j=0,l,n,q)	lmnqr	p(j=0)	kmnq	No
ilnqr	p(r|j=0,l,n,q)	lmnqr	p(m|j=0)	kmnq	No

FIG. 7. Given evidence $j = 0$ in the Hailfinder join tree of Figure 5, our system identifies the relevant and irrelevant incoming messages for the construction of every outgoing message.

the physical construction or the forwarding of $p(X_2|Y_2)$; otherwise, $p(X_1|Y_1)$ is irrelevant to $p(X_2|Y_2)$. By viewing each outgoing message $p(X_2|Y_2)$ sent out from a join tree node N_2 as a query posed to N_2's local directed acyclic graph D_{N_2}, the RelMsgId algorithm can determine which, if any, of the incoming messages $p(X_1|Y_1)$ are relevant.

Algorithm 2 RelMsgId $(A, p(v|Y), C)$

```
Input: $A$ - the set of distribution headings assigned to join
         tree node $N$,
         $p(v|Y)$ - the heading of an outgoing message sent
         from $N$ to a neighbor $N'$,
         $C$ - the headings of the incoming messages sent to
         $N$ from all neighbors except $N'$.
Output: classification of the headings in $C$ as relevant or
irrelevant with respect to constructing or forwarding $p(v|Y)$.
begin
Build the local directed acyclic graph $D_N$ uniquely defined
by $A$ and $C$ at join tree node $N$.
Apply the Prune algorithm on $D_N$ using $p(v|Y)$.
for each variable $v_i$ that was pruned
    if $p(v_i|P_i) \in C$
        Mark heading $p(v_i|P_i)$ as irrelevant.
for each variable $v_i$ that was not pruned
    if $v_i \neq v$ and $p(v_i|P_i) \in C$ and $v_i$ is a root evidence
    variable
        Mark heading $p(v_i|P_i)$ as irrelevant.
Mark all remaining unmarked headings in $C$ as relevant.
return $(C)$
end
```

Example 5. *Consider the message $p(i)$ sent from node ilnqr to its neighbor gijl in the Hailfinder join tree of Figure 3. RelMsgId builds the local directed acyclic graph defined by the incoming headings $p(i)$, $p(n|l)$, and $p(q|l,n)$ along with the assigned heading $p(r|i,l,n,q)$. Given this graph and query $p(i)$, the Prune algorithm removes the irrelevant variables l, n, q, and r. Thus, RelMsgId marks the incoming messages $p(n|l)$ and $p(q|l,n)$ from node lmnqqr as irrelevant to the forwarding of $p(i)$. On the contrary, the incoming message $p(i)$ from node fi is marked relevant. In the converse direction, consider message $p(m|j=0)$ sent from node lmnqr to its neighbor kmnq in Figure 5. RelMsgId builds the local directed acyclic graph defined by the incoming messages $p(j=0)$, $p(l|j=0)$, $p(r|j=0,l,n,q)$, and $p(m|l)$. The Prune algorithm removes irrelevant variables n, q, and r. Thus, RelMsgId marks $p(r|j=0,l,n,q)$ as irrelevant. Because j is a root evidence variable, RelMsgId marks $p(j=0)$ as irrelevant. On the other hand, the incoming messages $p(l|j=0)$ and $p(m|l)$ are marked as relevant.*

Given evidence $j = 0$ in the Hailfinder join tree, the screen-shot of our system in Figure 7 indicates whether or not an incoming message from node 1 is relevant at node 2 for the subsequent construction of a particular message from node 2 to node 3.

3.3. Physical Construction of Messages

A join tree node is allowed to physically build an outgoing message as soon as it has received all incoming messages that are relevant to its construction. Our approach uses VE to build distributions in computer memory. AR is not well suited for message construction in prioritized join tree propagation, because it constructs a new probability distribution for every child of the variable being eliminated [3]. As some of these new distributions can be deemed relevant and the rest irrelevant, we instead call VE to build the relevant distributions as needed. After all relevant messages have been constructed and sent, the irrelevant messages are built so that posterior probabilities can be computed when propagation finishes.

Example 6. *Consider how VE builds message $p(m|j=0)$ from node lmnqr to node kmnq in the Hailfinder join tree of Figure 5 when evidence $j=0$ is observed. By Example 5, only $p(m|l)$ and $p(l|j=0)$ are relevant to this task. To eliminate variable l, VE computes $p(m,l|j=0)=p(m|l)\cdot p(l|j=0)$, followed by $p(m|j=0)=\sum_l p(m,l|j=0)$. The probability table $p(m|j=0)$ is returned to lmnqr.*

Further to Example 6, while $p(r|j=0,l,n,q)$ is irrelevant with respect to subsequent message construction at node *lmnqr*, it is needed so that *lmnqr* can compute the posterior probability of its variables (given the evidence $j=0$) from the tables assigned to it together with those passed to it after propagation finishes. According to Figures 3 and 5, when propagation of the evidence $j=0$ finishes, the probability information at node *lmnqr* is more formally expressed as: $p(l,m,n,q,r,j=0)=p(j=0)\cdot p(l|j=0)\cdot p(m|l)\cdot p(n|m)\cdot p(q|m,n)\cdot p(r|j=0,l,n,q)$. The posterior probability $p(n|j=0)$, for instance, can be easily computed from this factorization.

The primary difference between our approach and Lazy AR is illustrated in Example 6, whereby our approach allows node *lmnqr* to physically build message $p(m|j=0)$ as soon as it has the received messages $p(l|j=0)$ and $p(m|l)$. On the contrary, the state-of-the-art method, Lazy AR, will force node *lmnqr* to wait for the reception of messages $p(j=0)$ and $p(r|j=0,l,n,q)$ as well, even though these two messages are not required in the physical construction of $p(m|j=0)$. This unnecessary delay in Example 6 is inherently built into the main philosophy of Lazy AR. Messages are classified as relevant or irrelevant in Step (ii) only after a join tree node collects all of its messages in Step (i). This immediately means that for any incoming message deemed irrelevant in Step (ii), Lazy AR has already forced the join tree node to wait for its physical construction prior to Step (i). On the contrary, in the three main steps of prioritized join tree propagation, the message headings identified in Step (i) are classified as relevant or irrelevant in Step (ii) before any physical computation takes place in computer memory by Step (iii). Therefore, by avoiding this type of unnecessary delay, our prioritized approach can perform Bayesian network inference faster than Lazy AR.

4. EXPERIMENTAL RESULTS

Here, we conduct an empirical comparison of Lazy AR and prioritized join tree propagation.

We first illustrate how prioritized messages can be exploited during parallel computation based on the following three assumptions: (i) there are two processors, denoted P_1 and P_2, each with a queue of messages to be built or forwarded; (ii) in Lazy AR, if a join tree node applies AR to physically build k messages $\{p(v_1|P_1), p(v_2|P_2), \ldots, p(v_k|P_k)\}$, then we place all k messages on the queue of same processor, because $p(v_1|P_1)$ is needed to build $p(v_2|P_2)$, and so on; (iii) in Lazy AR, the processor that finishes building or forwarding the messages sent from node N_1 to node N_2 will be used to build or forward the subsequent messages sent from N_2 to node N_3 ($N_3 \neq N_1$). The reason is that, in Lazy AR, if N_1 sends messages to N_2, then N_2 necessarily waits for all of N_1's messages before building or forwarding the subsequent messages to node N_3. In the next two examples, we often refer to messages with the sending and receiving nodes understood.

Example 7. *The optimal schedule in Lazy AR for propagating the eleven distributions from nodes dkoq, elm, and abcf towards node gijl in the real-world Hailfinder join tree of Figure 3 is:*

$$\begin{aligned} P_1 &: p(q|k,o), p(k), p(q|k,m), p(n|m), p(q|m,n),\\ &\quad p(n|l), p(q|l,n), p(i).\\ P_2 &: p(m|l), p(f), p(i). \end{aligned}$$

By (ii) and (iii), the eight messages $p(q|k,o)$, $p(k)$, $p(q|k,m)$, $p(n|m)$, $p(q|m,n)$, $p(n|l)$, $p(q|l,n)$, and $p(i)$ from node dkoq towards node gijl must necessarily be built in a serial fashion, say on processor P_1. This means that processor P_2 builds three messages, namely, $p(m|l)$, $p(f)$, and $p(i)$. Our optimal schedule is:

$$\begin{aligned} P_1 &: p(q|k,o), p(q|k,m), p(n|m), p(q|m,n), p(n|l), p(q|l,n).\\ P_2 &: p(k), p(m|l), p(f), p(i), p(i). \end{aligned}$$

Example 7 shows that, in our approach, when processor P_1 is building $p(q|k,o)$, processor P_2 is forwarding $p(k)$, because we mark $p(q|k,o)$ as irrelevant to the forwarding of $p(k)$. Moreover, $p(i)$ can be built and forwarded by P_2, because $p(f)$ is the only relevant message for building $p(i)$. Now, let us consider a more involved example.

Example 8. *In the Hailfinder join tree of Figure 5, the Lazy AR optimal schedule for propagating the thirteen distributions from nodes hxy, gst, abcf, and elm towards node kmnq is:*

$$\begin{aligned} P_1 &: p(h), p(j=0|i), p(j=0|i), p(l|i,j=0), p(j=0),\\ &\quad p(l|j=0), p(r|j=0,l,n,q), p(j=0). \qquad (3)\\ P_2 &: p(g), p(f), p(i), p(m|l), p(m|j=0). \end{aligned}$$

By (ii) and (iii), the first eight distributions sent from hxy towards kmnq must necessarily be built in a serial fashion, say on P_1. Therefore, the remaining five messages are placed in P_2's queue. Observe that even though there are two messages $p(j=0)$ and $p(m|j=0)$ sent from node lmnqr to kmnq,

TABLE 1. Description of real-world or benchmark Bayesian networks and the constructed join trees.

Bayesian network name	Number of Bayesian network variables	Number of join tree nodes	Number of variables in the largest join tree node
Alarm	37	27	5
Barley	48	36	8
CHD	11	5	6
Hailfinder	56	43	5
Insurance	27	18	8

TABLE 3. The performance of Lazy AR and our prioritized approach with nine percent evidence variables.

Bayesian network name	Number of evidence variables	Lazy AR	Prioritized approach	Speed-up over Lazy AR
Alarm	3	0.221	0.191	13.6%
Barley	4	1,257.560	1,017.651	19.1%
CHD	1	0.018	0.006	38.9%
Hailfinder	5	3.483	2.509	66.7%
Insurance	2	0.401	0.231	42.4%

$p(j = 0)$ can be simply forwarded by P_1 in Equation (3), while $p(m|j = 0)$ can be built by P_2, without the need of $p(j = 0)$ as shown in Example 3. The prioritized propagation optimal schedule is:

$$P_1 : p(h), p(j = 0|i), p(j = 0|i), p(l|i, j = 0),$$
$$p(l|j = 0), p(r|j = 0, l, n, q).$$
$$P_2 : p(g), p(f), p(i), p(j = 0), p(j = 0), p(m|l), p(m|j = 0).$$

The two important points in Example 8 are that in Equation (3), Lazy AR insists that $p(l|i, j = 0)$ be physically built before $p(j = 0)$ is constructed, even though $p(l|i, j = 0)$ is not needed in the construction of $p(j = 0)$. Moreover, for the second $p(j = 0)$ in Equation (3), Lazy AR requires $p(l|i, j = 0)$, $p(l|j = 0)$, and $p(r|j = 0, l, n, q)$ to be physically built even though they are not needed to forward $p(j = 0)$. These problems are avoided in our prioritized schedule. Examples 7 and 8 together illustrate that Lazy AR can impose unnecessary restrictions on when a distribution is built or forwarded. Therefore, Lazy AR often performs calculations in a serial manner when, in fact, they can be performed in a parallel fashion. By recognizing and removing these unnecessary restrictions, our prioritized approach necessarily runs faster.

It is also worth contrasting our work here with our previous work. In [3], we suggest applying the VE algorithm to build the messages in a Lazy AR schedule rather than applying the AR algorithm to construct the messages as Lazy AR does. We explicitly demonstrated that the AR algorithm can build probability distributions that will not be passed as messages, nor are they needed in the construction of the messages that will be passed. Thus, although our previous work focused on how the messages are built, the work here focuses on when the messages are built. We now empirically demonstrate the time savings to be made by using a better schedule of when messages can be constructed.

In our empirical evaluation, both methods were implemented in the C++ programming language. The experiments were conducted on a 24-processor SGI Onyx2 graphics supercomputer. Each inference algorithm has two processors allocated for its sole usage. The evaluation was carried out on four real-world Bayesian networks, called Alarm, Barley, CHD and Hailfinder, as well as one benchmark Bayesian network known as Insurance. The corresponding join trees were built using the Netica system [17]. Table 1 describes each Bayesian network and its corresponding join tree.

Table 2 reports on Bayesian inference with no evidence variables. The running times in seconds are listed in the second and third columns for Lazy AR and prioritized join tree propagation, respectively. The last column shows the speed-up of our prioritized approach over Lazy AR, the average of which was 49.6%.

Next, the running times of Bayesian network inference involving evidence were measured. As shown in Tables 3 and 4, approximately nine percent and eighteen percent of the variables in each Bayesian network were randomly instantiated as evidence variables, respectively. Once again, prioritized join tree propagation was faster than Lazy AR in all Bayesian networks. In Table 3, the time saved ranged from 13.6 to 66.7% with an average of 36.1%. Similarly, in Table 4, the time saved ranged from 7.4 to 40.9% with an average of 21.7%.

5. CONCLUSIONS

This article is the first work to suggest the concept of prioritized join tree propagation. As illustrated in Examples 2 and 3, the motivation for this study is based on the observation

TABLE 2. The performance of Lazy AR and our prioritized approach without evidence variables.

Bayesian network name	Lazy AR	Prioritized approach	Speed-up over Lazy AR
Alarm	0.277	0.109	41.9%
Barley	13,323.781	4,133.274	69.0%
CHD	0.033	0.014	57.6%
Hailfinder	1.192	0.866	25.8%
Insurance	0.606	0.222	53.5%

TABLE 4. The performance of Lazy AR and our prioritized approach with eighteen percent evidence variables.

Bayesian network name	Number of evidence variables	Lazy AR	Prioritized approach	Speed-up over Lazy AR
Alarm	7	0.122	0.113	7.4%
Barley	9	30.889	27.857	9.8%
CHD	2	0.011	0.007	36.4%
Hailfinder	10	3.673	3.159	14.0%
Insurance	5	0.225	0.133	40.9%

that, during inference in real-world Bayesian networks, it is often the case that only some of the messages passed to a join tree node are actually needed in the physical construction of the subsequent probability distributions (messages) sent out from the node. Consequently, our approach first identifies the headings of all distributions to be propagated in the join tree, as illustrated by the screen-shot of Figure 6. With respect to each join tree node N, our system then labels each incoming message to N as either relevant or irrelevant to the physical construction of each message outgoing from N, as indicated by the screen-shot of Figure 7. Lastly, our system builds the relevant messages and then the irrelevant messages. As reported in Tables 2, 3, and 4, in all four real-world Bayesian networks and one benchmark Bayesian network and with varying amounts of evidence, prioritized join tree propagation finished faster than Lazy AR without exception. Future work includes the identification of those instances when the same distribution is to be constructed at multiple join tree nodes, as well as the development of a heuristic to estimate the node to best build the distribution.

Acknowledgments

The authors would like to thank anonymous reviewers for their contribution in making a more clear and focused paper.

REFERENCES

[1] B. Abramson, J. Brown, W. Edwards, A. Murphy, and R.L. Winkler, Hailfinder: A Bayesian system for forecasting severe weather, Int J Forecasting 12 (1996), 57–71.

[2] M. Baker and T.E. Boult, Pruning Bayesian networks for efficient computation, Proceedings of 6th Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, 1990, pp. 225–232.

[3] C.J. Butz and S. Hua, An improved LAZY-AR approach to Bayesian network inference, Proceedings of 19th Canadian Conference on Artificial Intelligence, Quebec City, QC, 2006, pp. 183–194.

[4] A. Cano, S. Moral, and A. Salmeron, LAZY evaluation in penniless propagation over join trees, Networks 39 (2002), 175–185.

[5] E. Castillo, J. Gutiérrez, and A. Hadi, A new method for efficient symbolic propagation in discrete Bayesian networks, Networks 28 (1996), 31–43.

[6] E. Castillo, J. Gutiérrez, and A. Hadi, Expert systems and probabilistic network models, Springer, New York, 1997.

[7] E. Castillo, A.S. Hadi, F. Jubete, and C. Solares, An expert system for coherent assessment of probabilities in multigraph models, Networks 33 (1999), 193–206.

[8] R.M. Chavez and G.F. Cooper, A randomized approximation algorithm for probabilistic inference on Bayesian belief networks, Networks 20 (1990), 661–685.

[9] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artif Intell 42 (1990), 393–405.

[10] P. Dagum and E. Horvitz, A Bayesian analysis of simulation algorithms for inference in belief networks, Networks 23 (1993), 499–516.

[11] D. Geiger, T. Verma, and J. Pearl, Identifying independence in Bayesian networks, Networks 20 (1990), 507–534.

[12] P. Hájek, T. Havránek, and R. Jiroušek, Uncertain information processing in expert systems, CRC Press, Ann Arbor, 1992.

[13] F.V. Jensen, K.G. Olesen, and S.K. Andersen, An algebra of Bayesian belief universes for knowledge based systems, Networks 20 (1990), 637–659.

[14] A.L. Madsen, An empirical evaluation of possible variations of lazy propagation, Proceedings of 20th Conference on Uncertainty in Artificial Intelligence, Banff, AB, 2004, pp. 366–373.

[15] A.L. Madsen, Variations over the message computation algorithm of lazy propagation, IEEE Trans Sys Man Cybern, B 36 (2006), 636–648.

[16] A.L. Madsen and F.V. Jensen, Parallelization of inference in Bayesian networks, Research report R-99-5002, Aalborg University, 1999, Denmark.

[17] Norsys Software Corp., Netica software, webpage, Available at: http://www.norsys.com/.

[18] S. Olmsted, On representing and solving decision problems, PhD Thesis, Department of Engineering Economic Systems, Stanford University, Stanford, California, 1983.

[19] J. Pearl, Probabilistic reasoning in intelligent systems: Networks of plausible inference, Morgan Kaufmann, San Francisco, 1988.

[20] C. A. de B. Pereira and R. E. Barlow, Medical diagnosis using influence diagrams, Networks 20 (1990), 565–577.

[21] R. Shachter, Evaluating influence diagrams, Oper Res 34 (1986), 871–882.

[22] R. Shachter, An ordered examination of influence diagrams, Networks 20 (1990), 535–563.

[23] G. Shafer, Probabilistic expert systems, SIAM, Philadelphia, 1996.

[24] D.J. Spiegelhalter and S.L. Lauritzen, Sequential updating of conditional probabilities on directed graphical structures, Networks 20 (1990), 579–605.

[25] S.K.M. Wong, C.J. Butz, and D. Wu, On the implication problem for probabilistic conditional independency, IEEE Trans Sys Man Cybern A 30 (2000), 785–805.

[26] Y. Xiang, Verification of DAG structures in cooperative belief network-based multiagent systems, Networks 31 (1998), 183–191.

[27] Y. Xiang, Cooperative triangulation in MSBNs without revealing subnet structures, Networks 37 (2001), 53–65.

[28] N.L. Zhang and D. Poole, A simple approach to Bayesian network computations, Proceedings of 10th Canadian Conference on Artificial Intelligence, Banff, AB, 1994, pp. 171–178.