Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Bayesian Network Inference with Simple Propagation

Cory J. Butz
butz@cs.uregina.ca
University of Regina
Canada

Jhonatan S. Oliveira
oliveira@cs.uregina.ca
University of Regina
Canada

André E. dos Santos
dossantos@cs.uregina.ca
University of Regina
Canada

Anders L. Madsen
anders@hugin.com
HUGIN EXPERT A/S
Aalborg University
Denmark

Abstract

We propose *Simple Propagation* (SP) as a new join tree propagation algorithm for exact inference in discrete Bayesian networks. We establish the correctness of SP. The striking feature of SP is that its message construction exploits the factorization of potentials at a sending node, but without the overhead of building and examining graphs as done in *Lazy Propagation* (LP). Experimental results on numerous benchmark Bayesian networks show that SP is often faster than LP.

Introduction

Join tree propagation (JTP) is central to the theory and practice of probabilistic expert systems (Shafer 1996). Here, exact inference in a discrete *Bayesian network* (BN) (Pearl 1988) is conducted on a secondary structure, called a join tree, built from the directed acyclic graph of a BN. Even though the computational and space complexity of JTP is exponential in the tree-width of the network, in general, we care not about the worse case, but about the cases we encounter in practice (Koller and Friedman 2009). For real-world BNs, several JTP approaches appear to work quite well. Given observed evidence, messages are systematically propagated such that posterior probabilities can be computed for every non-evidence variable. More specifically, each BN conditional probability table, having been updated with observed evidence, is assigned to precisely one join tree node containing its variables. Classical JTP algorithms (Shafer 1996) form one potential per node by multiplying together the tables at each node.

In contrast, *Lazy Propagation* (LP) (Madsen and Jensen 1999; Madsen 2004) keep a multiplicative factorization of potentials at each node. This allows LP to remove two kinds of irrelevant potentials during message construction. Irrelevant potentials involving *barren variables* (Madsen and Jensen 1999) are removed first from the factorization. Next, irrelevant potentials based on testing independencies induced by evidence are removed from the factorization. Here, a potential is irrelevant if and only if certain variables are separated in the moralization $\mathcal{G}_1^m$ of the *domain graph* (Madsen 2004) $\mathcal{G}_1$ built from the factorization. In the remaining relevant potentials, all variables not appearing in the separator need to be marginalized away. The order in which these variables are marginalized, called an *elimination ordering* (Madsen and Butz 2012), is determined by examining the moralization $\mathcal{G}_2^m$ of the domain graph $\mathcal{G}_2$ built from the factorization of relevant potentials. The resulting factorization is the message propagated.

In this paper, we propose *Simple Propagation* (SP) as a new JTP algorithm for exact inference in discrete BNs. SP consists of three simple steps. First, remove irrelevant potentials based on barren variables. Second, marginalize the relevant variables. More specifically, while the factorization at a sending node contains a potential with a non-evidence variable in the separator and another not in the separator, then the latter must be marginalized away. Third, propagate only those potentials exclusively containing variables in the separator. We establish the correctness of SP. Thus, SP is equivalent to LP, but without following key tenets of LP. SP never explicitly tests independencies, nor does it determine elimination orderings. This means SP saves the overhead of having to build and examine graphs. In experimental results on 28 benchmark cases, SP is faster than LP in 18 cases, ties LP in 5 cases, and is slower than LP in 5 cases.

Background

Let $U = \{v_1, v_2, \ldots, v_n\}$ be a finite set of variables, each with a finite domain, and V be the domain of U. A *potential* on V is a function ϕ such that $\phi(v) \geq 0$ for each $v \in V$, and at least one $\phi(v) > 0$. Henceforth, we say ϕ is on U instead of V. A *joint probability distribution* is a potential P on U, denoted $P(U)$, that sums to one. For disjoint $X, Y \subseteq U$, a *conditional probability table* (CPT) $P(X|Y)$ is a potential over $X \cup Y$ that sums to one for each value y of Y. For simplified notation, $\{v_1, v_2, \ldots, v_n\}$ may be written as $v_1 v_2 \cdots v_n$, and $X \cup Y$ as XY.

A *Bayesian network* (BN) (Pearl 1988) is a *directed acyclic graph* (DAG) $\mathcal{B}$ on U together with CPTs $P(v_1|Pa(v_1))$, $P(v_2|Pa(v_2))$, ..., $P(v_n|Pa(v_n))$, where $Pa(v_i)$ denotes the parents (immediate predecessors) of v_i in $\mathcal{B}$. For example, Figure 1 depicts a BN, where CPTs $P(a), P(b|a), \ldots, P(m|g,l)$ are not shown. We call $\mathcal{B}$ a BN, if no confusion arises. The product of the CPTs for $\mathcal{B}$ on U is a joint probability distribution $P(U)$.

The *conditional independence* (Pearl 1988) of X and Z

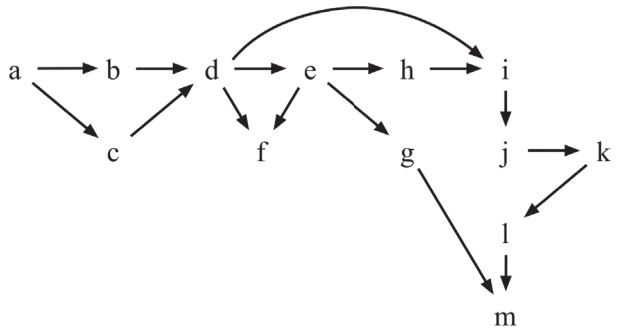


Figure 1: A BN extended from (Madsen and Jensen 1999).

given Y holding in $P(U)$ is denoted $I(X, Y, Z)$, where X, Y, and Z are pairwise disjoint subsets of U. If needed, the property that $I(X, Y, Z)$ is equivalent to $I(X-Y, Y, Z-Y)$ (Pearl 1988) can be applied to make the three sets pairwise disjoint; otherwise, $I(X, Y, Z)$ is not well-formed.

A *join tree* (Shafer 1996) 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 *separator* (Shafer 1996) S between any two neighbouring nodes N_i and N_j is $S = N_i \cap N_j$. A DAG $\mathcal{B}$ can be converted into a join tree via the moralization and triangulation procedures. The *moralization* (Pearl 1988) $\mathcal{B}^m$ of $\mathcal{B}$ is obtained by adding undirected edges between all pairs of vertices with a common child and then dropping directionality. An undirected graph is *triangulated* (Pearl 1988), if each cycle of length four or more has an edge between two nonadjacent variables in the cycle. Each *maximal clique* (complete subgraph) (Pearl 1988) of the triangulated graph is represented by a node in the join tree.

Prior to inference, the BN CPTs are updated by deleting all rows (configurations) disagreeing with the observed evidence $E = e$, if any. Lastly, each CPT $P(v_i|Pa(v_i))$ is assigned to exactly one join tree node N containing the variables $v_i \cup Pa(v_i)$. We now say the join tree is *initialized*.

Example 1. *One possible join tree for the BN $\mathcal{B}$ in Figure 1 has the following three nodes:*

$$\begin{aligned} N_1 &= \{a, b, c\}, \\ N_2 &= \{b, c, d, e, f, g, h, i, j, l, m\}, \text{and} \\ N_3 &= \{i, j, k, l, m\}. \end{aligned}$$

Let the observed evidence in $\mathcal{B}$ be $d = 0$. Those CPTs containing d are updated by keeping only those rows with $d = 0$. Assigning the CPTs to N_1, N_2, and N_3 can yield the following respective factorizations $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$:

$$\begin{aligned} \mathcal{F}_1 &= \{P(a), P(b|a), P(c|a)\}, \\ \mathcal{F}_2 &= \{P(d = 0|b, c), P(e|d = 0), P(f|d = 0, e), \\ &\quad P(g|e), P(h|e), P(i|d = 0, h), P(j|i), P(m|g, l)\}, \\ \mathcal{F}_3 &= \{P(k|j), P(l|k)\}. \end{aligned}$$

Rather than multiplying together the CPTs at each node, *Lazy Propagation* (LP) (Madsen and Jensen 1999) maintains a multiplicative factorization. Doing so allows LP in message construction to safely remove (without performing numerical computation) two kinds of irrelevant potentials from the factorization. The first kind of irrelevant potential is based upon the notion of *barren* variables (Madsen and Jensen 1999). Given a factorization $\mathcal{F}$, and with respect to a separator S, the procedure REMOVEBARREN recursively removes each potential $\phi(W|Z)$, if no variable in W appears in another potential in $\mathcal{F}$ and $W \cap S = \emptyset$. Detecting the second kind of irrelevant potential is more involved.

LP tests separation in a graph, a process that reflects testing independencies induced by evidence, to safely remove irrelevant potentials from the factorization. A potential is irrelevant if and only if the corresponding separation holds in the graph. The original LP (Madsen and Jensen 1999) applied *d-separation* (Pearl 1988) in the given BN $\mathcal{B}$. The extensions of LP (Madsen 2004) test separation in the domain graph constructed from the factorization under consideration. The *domain graph* (Madsen 2004) of a potential $\phi(W|Z)$ is a graph with undirected edges between $v_i, v_j \in W$ and directed edges from each $v_k \in Z$ to each $v_l \in W$. The domain graph of a set of potentials is defined in the obvious way.

All remaining potentials are relevant. Now, all variables not appearing in the separator need to be marginalized. The order in which these variables are marginalized, called an *elimination ordering* (Madsen and Butz 2012), is determined by examining the moralization $\mathcal{G}_2^m$ of the domain graph $\mathcal{G}_2$ built from the factorization of relevant potentials. Note that the moralization of a domain graph also adds undirected edges between all pairs of vertices with children connected by an undirected path before dropping directionality. *Fill-in weight* is one heuristic for finding good elimination orderings in LP (Madsen and Butz 2012).

Based upon the eloquent discussion of JTP in (Shafer 1996), we describe LP's execution on an initialized join tree with a chosen root using five rules:

Rule 1. Augment each node N with evidence variables E.

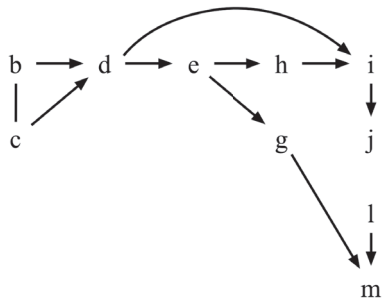
Rule 2. Each non-root node waits to send its message to a given neighbour until it has received messages from all its other neighbours.

Rule 3. The root waits to send messages to its neighbours until it has received messages from them all.

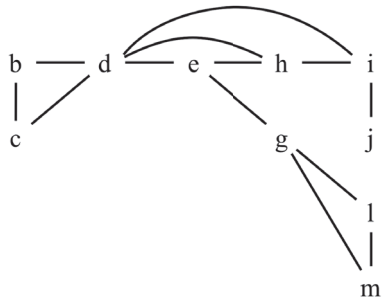
Rule 4. When a node is ready to compute its message to a particular neighbour, it calls $\text{MC}(\mathcal{F}, S, E)$ in Algorithm 1, where $\mathcal{F}$ is the set of assigned CPTs and all incoming messages received, S is the separator, and E is the set of evidence variables.

Rule 5. If a separator has received a message m from one of its two nodes, it sets the other message m' to be $m' = m' - m$, when it is received.

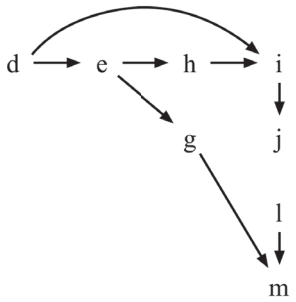
Example 2. *In the initialized join tree in Example 1, Rule 1 of LP sets $N_1 = N_1 \cup \{d\}$, $N_2 = N_2 \cup \{d\}$, and $N_3 = N_3 \cup \{d\}$, since the evidence is $d = 0$. If N_3 is chosen*



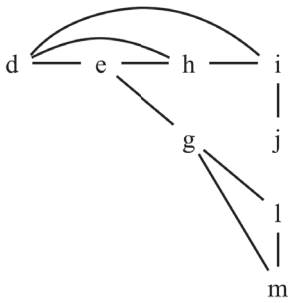
(i) LP constructs the domain graph $\mathcal{G}_1$ of $\mathcal{F}$ in (3).



(ii) LP uses separation analysis in the moralization $\mathcal{G}_1^m$ of $\mathcal{G}_1$ in (i) to determine irrelevant potentials in $\mathcal{F}$ of (3).



(iii) LP builds the domain graph $\mathcal{G}_2$ of $\mathcal{F}$ in (4) formed by relevant potentials.



(iv) LP uses the moralization $\mathcal{G}_2^m$ of $\mathcal{G}_2$ in (iii) to obtain an elimination ordering for variables e, g, and h.

Figure 2: LP's use of graphs to test independencies and determine elimination orderings at node N_2 in Example 2.

as root, Rules 2 and 3 mean that messages will be passed as follows: m_1 from N_1 to N_2; m_2 from N_2 to N_3; m_3 from N_3 to N_2; and, m_4 from N_2 to N_1. The discussion becomes interesting after N_1 sends N_2 its message m_1 computed as:

$$P(b,c) = \sum_a P(a,b,c) = \sum_a P(a) \cdot P(b|a) \cdot P(c|a). \quad (1)$$

N_2 calls MC$(\mathcal{F}, S, E)$ *to construct its message m_2 to N_3, where $S = \{d, i, j, l, m\}$, $E = \{d\}$, and $\mathcal{F}$ is:*

$$\mathcal{F} \quad = \quad \mathcal{F}_2 \cup \{P(b,c)\}. \quad (2)$$

In line 2, REMOVEBARREN *removes $P(f|d = 0, e)$ from $\mathcal{F}$, since variable f appears in no other potential of $\mathcal{F}$ and $\{f\} \cap S = \emptyset$, giving*

$$\mathcal{F} = \{P(b,c), P(d=0|b,c), P(e|d=0), P(g|e), P(h|e), P(i|d=0,h), P(j|i), P(m|g,l)\}. \quad (3)$$

Next, in line 3, the domain graph $\mathcal{G}_1$ in Figure 2 (i) is constructed from $\mathcal{F}$. In line 4, the moralization $\mathcal{G}_1^m$ of $\mathcal{G}_1$ is depicted in Figure 2 (ii). In line 5, considering potential $P(b,c)$, LP tests $I(bc, d, ijlm)$ in $\mathcal{G}_1^m$, namely, whether evidence variable d separates the variables b and c in $P(b,c)$ from the separator variables S. Since this separation in $\mathcal{G}_1^m$ holds, LP safely removes irrelevant potential $P(b,c)$ from $\mathcal{F}$, as seen in line 7. Similarly, for potential $P(d = 0|b,c)$, $I(bc, d, ijlm)$ holds in $\mathcal{G}_1^m$ meaning that $P(d = 0|b,c)$ also is removed from $\mathcal{F}$ in line 7. For the other six potentials in $\mathcal{F}$, it can be verified that $I(e, d, ijlm)$, $I(eg, d, ijlm)$, $I(eh, d, ijlm)$, $I(ih, d, ijlm)$, $I(ij, d, ijlm)$, and $I(glm, d, ijlm)$ do not hold in $\mathcal{G}_1^m$ by separation analysis or are not well-formed.

The remainder of MC *is to compute $\sum_{e,g,h} \mathcal{F}$, where*

$$\mathcal{F} \quad = \quad \{P(e|d=0), P(g|e), P(h|e), P(i|d=0,h), P(j|i), P(m|g,l)\}. \quad (4)$$

In line 8, the domain graph $\mathcal{G}_2$ of $\mathcal{F}$ is depicted in Figure 2 (iii). The moralization $\mathcal{G}_2^m$ of $\mathcal{G}_2$, illustrated in Figure 2 (iv), is obtained in line 9. Here, the fill-in weight heuristic can yield the elimination ordering $\sigma = (h, e, g)$. Assuming binary variables, the fill-in weight of h in Figure 2 (iv) is 4, since the weight of the edge (e, i) to be added is $2 \cdot 2 = 4$.

Algorithm 1 Message Construction.

```
1:  procedure MC(F, S, E)
2:      F = REMOVEBARREN(F, S)
3:      Construct the domain graph G_1 of F
4:      Construct the moralization G_1^m of G_1
5:      for each potential φ(X) in F do
6:          if I(X, E, S) holds in G_1^m then
7:              F = F − {φ(X)}
8:      Construct the domain graph G_2 of F
9:      Construct the moralization G_2^m of G_2
10:     Determine an elimination ordering σ in G_2^m
11:     for each v in σ do
12:         F = SUMOUT(v, F)
13:     return F
```

Following $\sigma = (h, e, g)$, LP computes in lines 11 and 12:

$$P(i|d=0,e) = \sum_h P(h|e) \cdot P(i|d=0,h),$$

$$P(g,i|d=0) = \sum_e P(e|d=0) \cdot P(g|e) \cdot P(i|d=0,e),$$

and

$$P(i,m|d=0,l) \quad = \quad \sum_g P(g,i|d=0) \cdot P(m|g,l).$$

LP sends message $m_2 = \mathcal{F} = \{P(j|i), P(i,m|d=0,l)\}$ to N_3 in line 13. This concludes the inward phase. In the outward phase, it can be shown that N_3 sends $P(l|j)$ to N_2, while N_2 sends $P(d=0|b,c)$ to N_1.

The important point of Example 2 is that, as illustrated in Figure 2, LP's message construction involves building domain graphs and their moralizations to test independencies and to determine elimination orderings.

Simple Propagation

Simple propagation (SP) is a novel JTP algorithm, since its message construction, called *Simple Message Construction* (SMC) and given in Algorithm 2, exploits the factorization of potentials without building and examining graphs. SP follows the same rules as LP, except it calls SMC in Rule 4.

Algorithm 2 Simple Message Construction.

```
1: procedure SMC(F, S, E)
2:     F = RemoveBarren(F, S)
3:     while ∃ φ(X) ∈ F with v ∉ S − E, v' ∈ S − E do
4:         F = SumOut(v, F)
5:     return {φ(X) ∈ F | X ⊆ S}
```

In the following example, we emphasize line 3 of SMC. By "one in, one out," we mean a potential in $\mathcal{F}$ has a non-evidence variable in the separator and another non-evidence variable not in the separator.

Example 3. *Let N_3 be the root of the initialized join tree in Example 1. Rule 1 augments each node with d. By Rule 2, N_1 is ready to send its message to N_2. By Rule 4, N_1 calls $\text{SMC}(\mathcal{F}, S, E)$ with $\mathcal{F} = \{P(a), P(b|a), P(c|a)\}$, $S = \{b, c, d\}$, and $E = \{d\}$. In line 2, no potentials are removed by* RemoveBarren. *In line 3, potential $P(b|a)$ contains non-evidence variable b in the separator and non-evidence variable a not in the separator. In line 4, SP eliminates a as in (1). Hence, N_1 sends message $P(b,c)$ to N_2.*

Node N_2 calls $\text{SMC}(\mathcal{F}, S, E)$ to compute its message to node N_3, where factorization $\mathcal{F}$ is in (2), separator $S = \{d, i, j, l, m\}$, and $E = \{d\}$. In line 2, RemoveBarren *removes $P(f|d=0,e)$ from $\mathcal{F}$, giving (3).*

In order to aid the reader in visualizing the "one in, one out" feature of SP in line 3, we graphically depict in Figure 3 the potentials in $\mathcal{F}$ of (3) based upon the techniques described in (Butz, Oliveira, and dos Santos 2015). For example, $P(b,c)$ is represented as a closed curve around white variables b and c, while $P(m|g,l)$ is represented as closed curve around white variable m and black variables g and l. In $P(m|g,l)$, for instance, m and l are non-evidence variables in the separator and g is a non-evidence variable not in the separator, as can be seen in Figure 3. Thus, in line 4, g is eliminated as

$$P(m|e,l) \;=\; \sum_g P(g|e) \cdot P(m|g,l),$$

yielding the new factorization:

$$\begin{aligned}\mathcal{F} \quad = \quad & \{P(b,c), P(d=0|b,c), P(e|d=0), P(h|e),\\ & P(i|d=0,h), P(j|i), P(m|e,l)\}. \end{aligned} \tag{5}$$

In the potential, say $P(m|e,l)$, m and l are non-evidence variables in the separator and e is a non-evidence variable not in the separator. Thus, SP eliminates e as

$$P(h,m|d=0,l) \;=\; \sum_e P(e|d=0) \cdot P(h|e) \cdot P(m|e,l),$$

giving the following factorization:

$$\begin{aligned}\mathcal{F} \quad = \quad & \{P(b,c), P(d=0|b,c), P(i|d=0,h),\\ & P(j|i), P(h,m|d=0,l)\}. \end{aligned}$$

Now, in $P(i|d=0,h)$, for instance, i is a non-evidence variable in the separator and h is a non-evidence variable not in the separator. Thus, SP eliminates h as

$$P(i,m|d=0,l) \;=\; \sum_h P(i|d=0,h) \cdot P(h,m|d=0,l),$$

leaving:

$$\mathcal{F} \;=\; \{P(b,c), P(d=0|b,c), P(j|i), P(i,m|d=0,l)\}.$$

SP sends message $m_2 = \{P(j|i), P(i,m|d=0,l)\}$ to N_3 in line 5. The outward phase is not described.

The key point of Example 3 is that SP message construction does not involve building and examining graphs.

Correctness

Suppose a join tree node N is ready to send its message to a neighbour node N' and calls $\text{SMC}(\mathcal{F}, S, E)$. After the call RemoveBarren$(\mathcal{F})$ in line 2, we partition $\mathcal{F}$ in line 3 into four disjoint sets in order to show correctness. Let

$$\mathcal{F}_{in} \quad = \quad \{\phi(X) \in \mathcal{F} \mid X \subseteq S\}$$

and

$$\begin{aligned}\mathcal{F}_{in-out} \quad = \quad & \{\phi(X) \in \mathcal{F} \mid X \text{ contains both}\\ & \text{a } v \notin S - E \text{ and a } v' \in S - E\}.\end{aligned}$$

$\mathcal{F}_{out}$ is defined as those potentials of $\mathcal{F}$ that were multiplied in the call $\text{SMC}(\mathcal{F}, S, E)$ by N, excluding those potentials defined in $\mathcal{F}_{in-out}$. Lastly,

$$\mathcal{F}_{rest} \quad = \quad \mathcal{F} - (\mathcal{F}_{in} \cup \mathcal{F}_{in-out} \cup \mathcal{F}_{out}).$$

For instance, in SP, recall N_2's call $\text{SMC}(\mathcal{F}, S, E)$ in Example 3. Here, $\mathcal{F}$ is given in (3). As illustrated by the shading in Figure 3,

$$\begin{aligned}\mathcal{F}_{in} \quad &= \quad \{P(j|i)\},\\ \mathcal{F}_{in-out} \quad &= \quad \{P(i|d=0,h), P(m|g,l)\},\\ \mathcal{F}_{out} \quad &= \quad \{P(e|d=0), P(g|e), P(h|e)\},\\ \mathcal{F}_{rest} \quad &= \quad \{P(b,c), P(d=0|b,c)\}.\end{aligned}$$

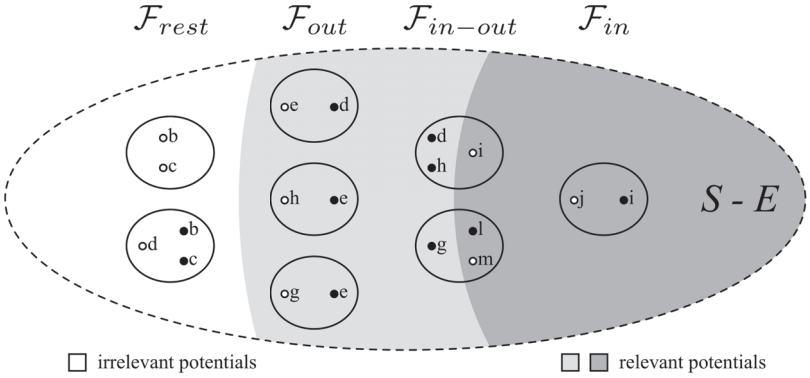


Figure 3: Visualizing the "one in, one out" property exploited in SP by graphically depicting the potentials in $\mathcal{F}$ of (3) at N_2. Variable g, for instance, needs to be eliminated, since potential $P(m|g,l)$ has l and m in $S-E$ and g not in $S-E$.

Let R be the set of variables appearing in the potentials of $\mathcal{F}_{rest}$ and let $O = (N-S)-R$. This means that O and R are disjoint and that

$$N - S \quad = \quad OR. \tag{6}$$

In Lemma 1, Lemma 2, and Theorem 1, we write $\prod_{\phi\in\mathcal{F}}\phi$ as $\mathcal{F}$ for simplified notation.

Lemma 1. *In the call* SMC$(\mathcal{F}, S, E)$ *by join tree node* N*, the output message* m *is:*

$$m \quad = \quad \mathcal{F}_{in} \cdot \sum_{O} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out}. \tag{7}$$

Proof. By definition, no potential in $\mathcal{F}_{rest}$ is multiplied in the call SMC$(\mathcal{F}, S, E)$. Thus, no potential of $\mathcal{F}_{rest}$ is included in m, since $R \cap S = \emptyset$ in line 5. On the contrary, line 5 means that every potential in $\mathcal{F}_{in}$ is necessarily included in m. Equation (7) immediately follows. □

Lemma 2. *The product of the potentials in* $\mathcal{F}_{rest}$ *is the marginal distribution* $P(R)$.

Proof. By definition, we have:

$$P(R) \quad = \quad \sum_{OS} \mathcal{F}_{in} \cdot \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \mathcal{F}_{rest} \cdot m',$$

where m' is the message sent from N' to N. Thus,

$$\begin{aligned} P(R) \quad &= \quad \sum_{S}\sum_{O} \mathcal{F}_{in} \cdot \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \mathcal{F}_{rest} \cdot m' \\ &= \quad \sum_{S} \mathcal{F}_{rest} \cdot m' \cdot \mathcal{F}_{in} \cdot \sum_{O} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out}. \end{aligned}$$

By Lemma 1, we obtain:

$$P(R) \quad = \quad \sum_{S} \mathcal{F}_{rest} \cdot m' \cdot m.$$

Since $P(S) = m' \cdot m$ (Shafer 1996), we have:

$$P(R) \quad = \quad \sum_{S} \mathcal{F}_{rest} \cdot P(S).$$

By manipulation,

$$\begin{aligned} P(R) \quad &= \quad \mathcal{F}_{rest} \cdot \sum_{S} P(S) \\ &= \quad \mathcal{F}_{rest} \cdot 1 \\ &= \quad \mathcal{F}_{rest}. \qquad \square \end{aligned}$$

We now show the main result of this paper, namely, that the SP algorithm is sound.

Theorem 1. *The call* SMC$(\mathcal{F}, S, E)$ *by a join tree node* N *is equal to* $\sum_{N-S}\mathcal{F}$.

Proof. Using (6), Lemma 1, and manipulation, we obtain:

$$\begin{aligned} \sum_{N-S}\mathcal{F} \quad &= \quad \sum_{N-S} \mathcal{F}_{in} \cdot \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \mathcal{F}_{rest} \\ &= \quad \sum_{OR} \mathcal{F}_{in} \cdot \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \mathcal{F}_{rest} \\ &= \quad \sum_{O}\sum_{R} \mathcal{F}_{in} \cdot \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \mathcal{F}_{rest} \\ &= \quad \mathcal{F}_{in} \cdot \sum_{O}\sum_{R} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \mathcal{F}_{rest} \\ &= \quad \mathcal{F}_{in} \cdot \sum_{O} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \sum_{R} \mathcal{F}_{rest}. \end{aligned}$$

By Lemma 2,

$$\begin{aligned} \sum_{N-S}\mathcal{F} \quad &= \quad \mathcal{F}_{in} \cdot \sum_{O} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot \sum_{R} P(R) \\ &= \quad \mathcal{F}_{in} \cdot \sum_{O} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \cdot 1 \\ &= \quad \mathcal{F}_{in} \cdot \sum_{O} \mathcal{F}_{in-out} \cdot \mathcal{F}_{out} \\ &= \quad m. \qquad \square \end{aligned}$$

Experimental Results

We report an empirical comparison between SP and LP. The experiments were performed on the 28 benchmark BNs listed in column 1 of Table 1. Column 2 shows the number

Table 1: Average time of LP and SP to propagate messages and compute posteriors over 100 runs in each BN.

BN	Vars	LP	SP	Savings
Water	32	0.06	0.05	17%
Oow	33	0.07	0.06	14%
Oow_Bas	33	0.04	0.03	25%
Mildew	35	0.05	0.04	20%
Oow_Solo	40	0.07	0.06	14%
Hkv2005	44	0.23	0.27	-17%
Barley	48	0.09	0.10	-11%
Kk	50	0.09	0.09	0%
Ship	50	0.16	0.17	-6%
Hailfinder	56	0.02	0.02	0%
Medianus	56	0.04	0.03	25%
3Nt	58	0.02	0.01	50%
Hepar_Ii	70	0.03	0.03	0%
Win95Pts	76	0.03	0.03	0%
System_V57	85	0.06	0.05	17%
Fwe_Model8	109	0.14	0.15	-7%
Pathfinder	109	0.12	0.11	8%
Adapt_T1	133	0.04	0.04	0%
Cc145	145	0.10	0.08	20%
Munin1	189	0.54	0.75	-39%
Andes	223	0.15	0.13	13%
Cc245	245	0.20	0.18	10%
Diabetes	413	0.34	0.31	9%
Adapt_T2	671	0.24	0.22	8%
Amirali	681	0.45	0.41	9%
Munin2	1003	0.49	0.45	8%
Munin4	1041	0.61	0.57	7%
Munin3	1044	0.66	0.64	3%

of variables in each BN. Join trees were generated using the *total weight* heuristic (Jensen 2014). The experimental analysis is performed using a Java 7 implementation running on a Linux Ubuntu server (kernel 2.6.38-16-server) with a four-core Intel Xeon(TM) E3-1270 Processor and 32 GB RAM.

For each network, 100 sets of evidence $E = e$ are randomly generated with both SP and LP using the same evidence. The computation time is measured as the elapsed (wall-clock) time for a full round of message passing (inward and outward) and computing posterior marginals $P(v|E = e)$ for each non-evidence variable. The average computation time is calculated over 100 runs and reported for LP and SP in columns 3 and 4 of Table 1, respectively.

Table 1 shows that out of 28 BNs, SP was faster in 18 cases, tied LP in 5 cases, and was slower than LP in 5 cases. SP tends to be faster than LP because SP does not spend time building and examining graphs. On the other hand, recall how Algorithm 1 is used in LP's Example 2. Lines 3-4 and 8-9 build the four graphs $\mathcal{G}_1$, $\mathcal{G}_1^m$, $\mathcal{G}_2$, and $\mathcal{G}_2^m$ in Figure 2, respectively. Lines 5-7 use separation analysis to test independencies in $\mathcal{G}_1^m$. Lastly, line 10 analyzes $\mathcal{G}_2^m$ in order to determine a good elimination ordering. In stark contrast, SP simply exploits the "one in, one out" property.

Conclusion

When a node N is ready to send a message to a neighbour sharing variables S, LP can be seen as focusing on $N - S$ to identify irrelevant potentials, and then working with the relevant potentials towards S. In comparison, SP can be viewed as starting at the separator S and working with the relevant potentials outward from S, thereby ignoring the irrelevant potentials. Roughly speaking, in Figure 3, LP works left-to-right, whereas SP works right-to-left.

Table 1 shows that SP tends to be faster than LP on optimal join trees built from a sampling of real-world and benchmarks BNs. It is possible, however, that LP will often be faster than SP on non-optimal join trees or even larger BNs, since there can be more variables to eliminate at each node, along with more possible elimination orderings. Future work will investigate the role of the particular join tree used in SP, as well as how to represent LP in DNs.

Acknowledgements

Research supported by NSERC Discovery Grant 238880.

References

Butz, C. J.; Oliveira, J. S.; and dos Santos, A. E. 2015. Darwinian networks. In *Proceedings of Twenty-Eighth Canadian Artificial Intelligence Conference*, 16–29.

Jensen, F. 2014. *HUGIN API Reference Manual - V. 8.1*.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Madsen, A. L., and Butz, C. J. 2012. On the importance of elimination heuristics in lazy propagation. In *Proceedings of Sixth European Workshop on Probabilistic Graphical Models*, 227–234.

Madsen, A. L., and Jensen, F. V. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence* 113(1-2):203–245.

Madsen, A. L. 2004. An empirical evaluation of possible variations of lazy propagation. In *Proceedings of Twentieth Uncertainty in Artificial Intelligence*, 366–373.

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

Shafer, G. 1996. *Probabilistic Expert Systems*, volume 67. Philadelphia: SIAM.