On Bayesian Network Inference with Simple Propagation

Cory J. Butz BUTZ@CS.UREGINA.CA
Jhonatan S. Oliveira OLIVEIRA@CS.UREGINA.CA
André E. dos Santos DOSSANTOS@CS.UREGINA.CA
Department of Computer Science
University of Regina
Regina

Anders L. Madsen ANDERS@HUGIN.COM
HUGIN EXPERT A/S
Department of Computer Science
Aalborg University
Aalborg

Abstract

Simple Propagation (SP) was recently proposed as a new join tree propagation algorithm for exact inference in discrete Bayesian networks and empirically shown to be faster than *Lazy Propagation* (LP) when applied on optimal (or close to) join trees built from real-world and benchmark Bayesian networks. This paper extends SP in two directions. First, we propose and empirically evaluate eight heuristics for determining elimination orderings in SP. Second, we show that the relevant potentials in SP are precisely those in LP.

Keywords: Bayesian networks; inference; join tree propagation; elimination orderings.

1. Introduction

Uncertainty can be managed by exact inference in discrete *Bayesian networks* (BNs) (Pearl, 1988; Koller and Friedman, 2009; Darwiche, 2009; Kjærulff and Madsen, 2013). A BN consists of a *directed acyclic graph* (DAG), where the vertices in the DAG represent variables in the problem domain, and a set of *conditional probability tables* (CPTs) matching the structure of the DAG. One approach to exact inference is *join tree propagation* (Shafer, 1996). Here, the DAG of a BN is transformed via the moralization and triangulation procedures into a *join tree* (Pearl, 1988). Each CPT, having been updated with observed evidence and called a *potential* (Shafer, 1996), is assigned to precisely one join tree node containing its variables. Messages are systematically propagated between neighbour nodes such that posterior probabilities can be computed for every non-evidence variable. Two join tree propagation algorithms of interest are Lazy Propagation and Simple Propagation.

Lazy Propagation (LP) (Madsen and Jensen, 1999) is a sophisticated join tree propagation algorithm. When a node is ready to construct a message, LP first distinguishes between *relevant* and *irrelevant* potentials using graphical methods. Next, with the relevant potentials, LP builds and uses graphs to determine *elimination orderings* (Koller and Friedman, 2009), an important practical consideration. *Simple Propagation* (SP) (Butz et al., 2016) was recently suggested as another join tree propagation algorithm. When a node N is ready to construct its message to a neighbour sharing variables S, SP recursively identifies those potentials at N with a “one in, one out” property, namely, the potential has at least one non-evidence variable in S and another not in S. SP then eliminates each "out" variable. SP is often faster than LP in experimental results involving optimal (or close to) join trees built from numerous real-world and benchmark BNs (Butz et al., 2016).

This paper extends SP in practical and theoretical directions. As there can be more than one potential satisfying the "one in, one out" property, the order in which these potentials are processed may affect SP's performance in practice. We propose and evaluate eight heuristics for determining elimination orderings in SP. For example, sorting potentials in increasing or decreasing order based on the number of variables in the potential or, more specifically, in increasing or decreasing order based on those variables of the potential that happen to be in the separator. Another four heuristics can be defined by considering variable domains rather than the number of variables in each case, respectively. Our experimental results, involving optimal (or close to) join trees built from 29 real-world and benchmark BNs, suggest that determining elimination orderings in SP often is wasteful. On the theoretical side, we establish that the relevant potentials in SP are exactly those in LP. We show this by utilizing the precise relationship between variables in LP's domain graphs and the "out" variables in SP. This is a stronger result than in (Butz et al., 2016), where it was only shown that SP is equivalent to LP, namely, the product of the messages are equal.

This paper is organized as follows. Section 2 gives background information. Elimination orderings are proposed in Section 3. In Section 4, we establish the equivalence between relevant potentials in SP and LP. Conclusions are drawn in Section 5.

2. 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 $v_1 v_2 \cdots v_n$, $X \cup Y$ as XY, and $\phi_k(X)$ as ϕ_k.

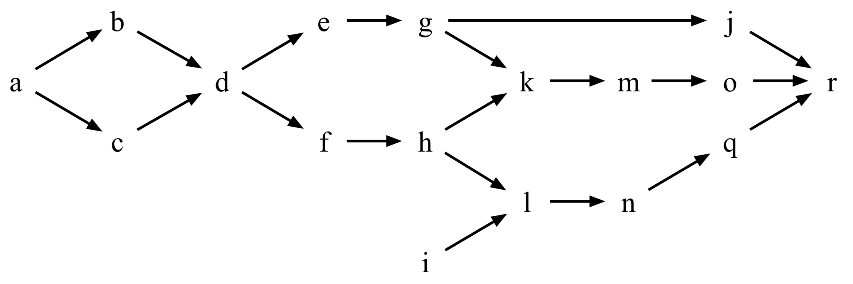

Figure 1: A BN extended from (Madsen and Jensen, 1999).

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 $P(a)$, $P(b|a)$, ..., $P(r|j, o, q)$ 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 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* (Pearl, 1988) 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* (Lauritzen and Spiegelhalter, 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* (Kjærulff, 1990), 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.

The BN CPTs are updated by deleting all 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 $v_i \cup Pa(v_i)$. Now we say the join tree is *initialized*.

Example 1 *Recall the BN $\mathcal{B}$ in Figure 1, where all variables are binary. Consider the join tree for $\mathcal{B}$ with three nodes: $N_1 = \{a, b, c\}$, $N_2 = \{b, c, d, e, f, g, h, i, j, k, l, m, n, o, q\}$, and $N_3 = \{j, o, q, r\}$. Let the observed evidence in $\mathcal{B}$ be $d = 0$ and $n = 1$. Those CPTs containing d and n are updated by keeping only those configurations with $d = 0$ and $n = 1$. Assigning the CPTs to N_1, N_2, and N_3 can yield the following respective factorizations $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$:*

$$\begin{aligned}
\mathcal{F}_1 &= \{P(a), P(b|a), P(c|a)\}, \\
\mathcal{F}_2 &= \{P(d=0|b,c), P(e|d=0), P(f|d=0), P(g|e), P(h|f), P(i), P(j|g), P(k|g,h), \\
&\qquad P(l|h,i), P(m|k), P(n=1|l), P(o|m), P(q|n=1)\}, \\
\mathcal{F}_3 &= \{P(r|j,o,q)\}.
\end{aligned}$$

Simple propagation (SP) (Butz et al., 2016) is a novel JTP algorithm, since its message construction, called *Simple Message Construction* (SMC) and given in Algorithm 1, exploits the factorization of potentials without building and examining graphs. Doing so allows SP in message construction to safely remove (without performing numerical computation) some irrelevant potentials from the factorization. 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 $W \cap S = \emptyset$.

Algorithm 1 Simple Message Construction.

```
1: procedure SMC($\mathcal{F}$, $S$, $E$)
2:     $\mathcal{F}$ = REMOVEBARREN($\mathcal{F}$, $S$)
3:     while $\exists\ \phi(X) \in \mathcal{F}$ with two variables $v \notin S - E$ and $v' \in S - E$ do
4:         $\mathcal{F}$ = SUMOUT($v$, $\mathcal{F}$)
5:     return $\{\phi(X) \in \mathcal{F} \mid X \subseteq S\}$
```

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

Example 2 *Recall the initialized join tree of Example 1, where the evidence is $d = 0$ and $n = 1$. SP augments each node with the evidence variables d and n. If N_3 is chosen as root, 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. N_1 calls* $\text{SMC}(\mathcal{F}, S, E)$ *with $\mathcal{F} = \{P(a),\ P(b|a),\ P(c|a)\}$, $S = \{b, c, d, n\}$, and $E = \{d, n\}$. In line 2, no potentials are removed by* REMOVEBARREN. *In line 3, where $S - E = \{b, c\}$, 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 only needs to eliminate variable a as:*

$$P(b,c) \;=\; \sum_a P(a,b,c) \;=\; \sum_a P(a) \cdot P(b|a) \cdot P(c|a)\,. \tag{1}$$

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 $S = \{d, j, n, o, q\}$, $E = \{d, n\}$, and $\mathcal{F}$ is:*

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

In line 2, REMOVEBARREN *does not remove any potentials.*

In line 3, where $S - E = \{j, o, q\}$, $P(j|g)$, for instance, has non-evidence variable j in the separator and a non-evidence variable g not in the separator. Thus, in line 4, g is eliminated as

$$P(j,k|e,h) = \sum_g P(g|e) \cdot P(j|g) \cdot P(k|g,h)\,, \tag{3}$$

yielding the new factorization:

$$\begin{aligned}\mathcal{F} \;=\; &\{P(b,c), P(d=0|b,c), P(e|d=0), P(f|d=0), P(h|f), P(i), P(j,k|e,h),\\ &P(l|h,i), P(m|k), P(n=1|l), P(o|m), P(q|n=1)\}\,.\end{aligned} \tag{4}$$

In potential, say $P(j,k|e,h)$, j is a non-evidence variable in the separator and h is a non-evidence variable not in the separator. Thus, SP eliminates h, giving:

$$P(j,k,l|e,f,i) \;=\; \sum_h P(h|f) \cdot P(j,k|e,h) \cdot P(l|h,i), \tag{5}$$

yielding the following factorization:

$$\begin{aligned}\mathcal{F} \;=\; &\{P(b,c), P(d=0|b,c), P(e|d=0), P(f|d=0), P(i), P(j,k,l|e,f,i),\\ &P(m|k), P(n=1|l), P(o|m), P(q|n=1)\}.\end{aligned}$$

Similarly, in the potential, say $P(j,k,l|e,f,i)$, j is a non-evidence variable in the separator and f is a non-evidence variable not in the separator. Thus, SP eliminates f, yielding:

$$P(j,k,l|d=0,e,i) \;=\; \sum_f P(j,k,l|e,f,i) \cdot P(f|d=0)\,. \tag{6}$$

Similarly, SP eliminates i, e, k, l, and m. Thus, SP sends message

$$m_2 \;=\; \{P(j, n=1, o|d=0), P(q|n=1)\}$$

to N_3 in line 5.

In the outward phase, it can be verified that N_3 sends an empty message to N_2, while N_2 sends $P(d=0|b,c)$ to N_1.

3. Elimination Orderings in SP

When a node is ready to construct a message in SP, there may be more than one potential satisfying the "one in, one out" property. The order in which these potentials are processed may affect SP's performance in practice. In this section, we propose and empirically evaluate 8 heuristics for determining elimination orderings in SP. Four elimination ordering heuristics that can be used for evaluating a potential on X with respect to a separator S in SP are:

- *Increasing variables in X* (Inc X): order potentials satisfying the "one in, one out" property based on increasing number of variables in X.
- *Decreasing variables in X* (Dec X): order potentials satisfying the "one in, one out" property based on decreasing number of variables in X.
- *Increasing variables of X in S* (Inc in S): order potentials satisfying the "one in, one out" property based on increasing number of variables of both X and S.
- *Decreasing variables of X in S* (Dec in S): order potentials satisfying the "one in, one out" property based on decreasing number of variables of both X and S.

Four additional elimination ordering heuristics can be respectively defined by replacing the number of variables with their size, where size is the product of the domain cardinalities of the variables under consideration: *Increasing variables in X size* (Inc X Size), *Decreasing variables in X size* (Dec X Size), *Increasing variables of X in S size* (Inc in S Size), and *Decreasing variables of X in S size* (Dec in S Size).

Example 3 *Consider the factorization in (4) of Example 2. There are two potentials in $\mathcal{F}$ with the "one in, one out" property of line 3 of* SMC, *namely $P(j,k|e,h)$ and $p(o|m)$. The Inc X heuristic would order these potentials as $< P(o|m), P(j,k|e,h) >$, since the former has 2 variables, while the latter has 4. The Dec X heuristic would use the reverse ordering of the Inc X.*

The heuristics Inc S and Dec S would have no preference between the two potentials, since both have 1 variable in the separator S.

Considering the size of the potential, that of $P(o|m)$ is $2 \times 2 = 4$, while that of $P(j,k|e,h)$ is 16. Hence, the Inc X Size heuristic orders the potentials as $< P(o|m), P(j,k|e,h) >$, while Dec X Size uses the reverse ordering of Inc X Size.

Lastly, $P(j,k|e,h)$ and $P(o|m)$ have both 1 variable in S with size 2 when sorting by Inc in S Size or Dec in S Size.

We now report on an empirical evaluation of using elimination ordering heuristics in SP. The experiments were performed on the 29 benchmark BNs listed in column 1 of Table 1. Column 2 shows the number of variables in each BN. The heuristics used in our investigation are listed in the last eight columns of Table 1. Column 3 records the time taken to eliminate variables from those potentials satisfying the "one in, one out" property according to the arbitrary order that these potentials happens to have in computer memory. 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 generated at random with all heuristics using the same evidence on each network. The computation time in seconds is measured as the elapsed (wall-clock) time for a full round of message passing (inward and outward) and computing posteriors marginals $P(v|E = e)$ for each non-evidence variable in the network. The average computation time is calculated over 100 runs and reported in columns 3 to 11 of Table 1, respectively.

Note that although the heuristics attempt to order the potentials themselves, they do not order the variables to be marginalized from within a potential. For instance, variables k, e, and h will be eliminated arbitrarily from potential $P(j, k|e, h)$ in Example 3.

Table 1: Experimental results on 29 BNs eliminating variables in SP arbitrarily versus using 8 elimination ordering heuristics. Times in boldface indicate a unique winner for that BN.

BN	Vars	Arbitrary Order	Inc X	Dec X	Inc in S	Dec in S	Inc X Size	Dec X Size	Inc in S Size	Dec in S Size
Water	32	0.05	0.04	0.05	0.04	0.05	0.04	0.05	0.04	0.05
oow	33	0.06	0.05	0.06	0.05	0.06	0.06	0.06	0.06	0.06
oow_bas	33	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
Mildew	35	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
oow_solo	40	0.06	0.05	0.06	0.05	0.06	0.06	0.06	0.06	0.06
HKV	44	**0.02**	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
Barley	48	0.09	0.09	0.09	0.09	0.09	0.09	0.08	0.09	0.08
KK	50	0.08	0.08	0.08	0.08	0.08	0.08	0.07	0.08	0.07
ship	50	0.16	0.13	0.16	0.13	0.16	0.15	0.15	0.15	0.15
hailfinder	56	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
medianus	56	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
3nt	58	**0.01**	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Hepar_II	70	0.24	0.23	0.32	0.28	0.31	0.23	0.32	0.27	0.31
win95pts	76	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
system_v57	85	0.05	0.05	0.06	0.05	0.06	0.05	0.05	0.05	0.05
FEW	109	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.14
pathfinder	109	0.09	0.09	0.09	0.09	0.09	0.10	0.09	0.09	0.10
Adapt_T1	133	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
cc145	145	0.08	0.08	0.08	0.07	0.08	0.08	0.08	0.08	0.07
Munin1	189	**0.68**	0.72	0.72	0.84	0.90	0.71	0.69	0.76	0.84
andes	223	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
cc245	245	0.17	0.18	0.18	0.17	0.18	0.18	0.17	0.18	0.17
Diabetes	413	**0.27**	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28
Adapt_T2	671	0.19	0.19	0.19	0.20	0.20	0.19	0.20	0.20	0.20
Amirali	681	0.40	**0.39**	0.44	0.42	0.40	0.42	0.44	0.40	0.41
Munin2	1003	0.44	0.41	0.42	0.41	0.43	0.42	0.44	0.40	**0.39**
Munin4	1041	0.51	0.52	0.51	0.51	0.52	0.53	0.51	0.52	0.51
Munin3	1044	**0.53**	0.55	0.56	0.56	0.56	0.55	0.56	0.55	0.55
sacso	2371	**0.72**	0.75	0.74	0.74	0.74	0.75	0.74	0.74	0.75
Tied for first		13	16	11	17	9	12	14	11	13
Unique wins		**6**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Analysis of Table 1 shows that eliminating arbitrarily won 6 out of 8 times when there was a clear winner. In the other two cases, two heuristics each won once. These results suggest that SP does not require elimination orderings, provided that an optimal (or close to) join tree is built from the real-world BNs in Table 1. As SP's performance degrades dramatically when applied on non-optimal join trees (Madsen et al., 2016), elimination orderings may be useful to SP in these cases. This remains as future work.

4. Relevant Potentials in SP

In this section, we establish our second main contribution, namely, a one-to-one correspondence between the relevant potentials in SP and LP.

Rather than multiplying together the CPTs at each node, *Lazy Propagation* (LP) (Madsen and Jensen, 1999) maintains a multiplicative factorization. Thus, during message construction, LP safely removes (without performing numerical computation) two kinds of irrelevant potentials from the factorization. Similar to SMC, the first kind is irrelevant potential of barren variables. However, detecting the second kind of irrelevant potential is more involved.

In principle, 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, 2010) 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. Afterwards, 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 of the domain graph built from the factorization of relevant potentials. 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). More formally, LP uses Algorithm 2, called Message Construction (MC), when a node is ready to construct its message to a neighbour. LP passes messages in the same sequence as SP, but calls MC instead of SMC.

Algorithm 2 Message Construction.

```
1:  procedure MC($\mathcal{F}$, $S$, $E$)
2:      $\mathcal{F}$ = REMOVEBARREN($\mathcal{F}$, $S$)
3:      Construct the domain graph $\mathcal{G}_1$ of $\mathcal{F}$
4:      Construct the moralization $\mathcal{G}_1^m$ of $\mathcal{G}_1$
5:      for each potential $\phi(X)$ in $\mathcal{F}$ do
6:          if $I(X, E, S)$ holds in $\mathcal{G}_1^m$ then
7:              $\mathcal{F} = \mathcal{F} - \{\phi(X)\}$
8:      Determine an elimination ordering $\sigma$
9:      for each $v$ in $\sigma$ do
10:         $\mathcal{F}$ = SUMOUT($v$, $\mathcal{F}$)
11:     return $\mathcal{F}$
```

We use Example 4 to illustrate how LP works.

Example 4 *Let N_3 be the root of the initialized join tree in Example 1. LP augments each node with the evidence variables d and n. N_1 calls $\mathrm{MC}(\mathcal{F}, S, E)$ with $\mathcal{F} = \{P(a), P(b|a), P(c|a)\}$, $S = \{b, c, d, n\}$, and $E = \{d, n\}$. LP eliminates variable a as in (1). Hence, N_1 sends message $P(b, c)$ to N_2.*

Node N_2 calls $\mathrm{MC}(\mathcal{F}, S, E)$ to compute its message to node N_3, where factorization $\mathcal{F}$ is in (2), separator $S = \{d, j, n, o, q\}$, and $E = \{d, n\}$. In line 2, no potential is removed.

Next, in line 3, the domain graph $\mathcal{G}_1$ is constructed from $\mathcal{F}$. In line 4, the moralization $\mathcal{G}_1^m$ of $\mathcal{G}_1$ is illustrated in Figure 2. In line 5, considering potential $P(b, c)$, LP tests $I(bc, dn, joq)$ in $\mathcal{G}_1^m$, namely, whether evidence variables d and n separate 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 $\mathcal{F}$, in line 7. Similarly, for potential $P(d = 0|b, c)$, $I(bc, dn, joq)$ 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 twelve potentials in $\mathcal{F}$, it can be verified that $I(e, dn, joq)$, $I(eg, dn, joq)$, $I(f, dn, joq)$, $I(fh, dn, joq)$, $I(ghk, dn, joq)$, $I(i, dn, joq)$, $I(hil, dn, joq)$, $I(km, dn, joq)$, $I(l, dn, joq)$, $I(gj, dn, joq)$, $I(mo, dn, joq)$, and $I(q, dn, joq)$, either do not hold in $\mathcal{G}_1^m$ by separation or are not well-formed.

The remainder is to compute $\sum_{i,e,f,g,h,k,l,m} \mathcal{F}$ using lines 8-10 of MC, *where*

$$\begin{aligned}\mathcal{F} \;=\; & \{P(e|d=0), P(f|d=0), P(g|e), P(h|f), P(i), P(j|g), \\ & P(k|g,h), P(l|h,i), P(m|k), P(n=1|l), P(o|m), P(q|n=1)\}\,. \end{aligned} \tag{7}$$

Next, the fill-in weight heuristic can yield the elimination ordering $\sigma = (i, e, f, g, h, k, l, m)$. Following σ, LP computes in lines 9 and 10:

$$\begin{aligned}
P(l|h) &= \sum_i P(i) \cdot P(l|h, i), \\
P(g|d=0) &= \sum_e P(g|e) \cdot P(e|d=0), \\
P(h|d=0) &= \sum_f P(f|d=0) \cdot P(h|f), \\
P(j,k|d=0,h) &= \sum_g P(g|d=0) \cdot P(k|g,h) \cdot P(j|g), \\
P(j,k,l|d=0) &= \sum_h P(h|d=0) \cdot P(j,k|d=0,h) \cdot P(l|h), \\
P(j,l,m|d=0) &= \sum_k P(j,k,l|d=0) \cdot P(m|k), \\
P(j,m,n=1|d=0) &= \sum_l P(j,l,m|d=0) \cdot P(n=1|l), \\
P(j,n=1,o|d=0) &= \sum_m P(j,m,n=1|d=0) \cdot P(o|m).
\end{aligned}$$

LP sends message $m_2 = \{P(j, n = 1, o|d = 0), P(q|n = 1)\}$ to N_3 in line 11. This concludes the inward phase. The outward phase is not described, but proceeds in the same manner.

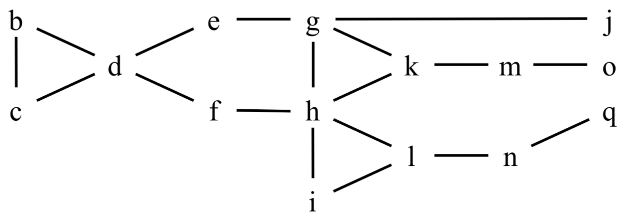

Figure 2: The moralization $\mathcal{G}_1^m$ of the domain graph at node N_2.

The main result of this section, given next in Theorem 1, establishes a one-to-one correspondence between the relevant potentials in SP and LP.

Theorem 1 *Given evidence E and factorization $\mathcal{F}$, suppose join tree node N is ready to construct its message to a neighbour sharing variables S. A potential $\phi(X) \in \mathcal{F}$ is relevant in the call* $\mathrm{MC}(\mathcal{F}, S, E)$ *in LP if and only if $\phi(X)$ is relevant in the call* $\mathrm{SMC}(\mathcal{F}, S, E)$ *in SP.*

Proof ($\Rightarrow$) Suppose LP determines potential $\phi(X)$ is relevant for message construction. By definition, S and X are not separated by evidence E in G, where G is the moralization of the domain graph of the factorization $\mathcal{F}$ of potentials at N. Hence, there exists a path in G of non-evidence variables and with a minimum number of edges from a variable in S to a variable in X. Let $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ denote this path, where $v_1 \in S$ and $v_k \in X$. Since the path has a minimum number of edges, the $k-1$ edges in the path represent $k-1$ distinct potentials, denoted $\phi_1, \phi_2, \ldots, \phi_{k-1}$; otherwise, if two edges come from the same potential, then there exists another path from S to X with fewer edges. Again, since the path has a minimum number of edges, none of $v_2, v_3, \ldots, v_k$ are members of S. Let us now consider SP. Potential ϕ_1 contains non-evidence variable $v_1 \in S$ and non-evidence variable $v_2 \notin S$. By construction, SP eliminates v_2. This involves multiplying ϕ_2, since ϕ_2 contains v_2, and marginalizing v_2, a process which leaves a potential containing both v_1 and v_3. Again, by construction, SP eliminates v_3. This elimination involves potential ϕ_3, since ϕ_3 contains v_3, and results in a potential containing both v_1 and v_4. This process repeats until the resulting potential contains both v_1 and v_k. Once again, by construction, SP eliminates v_k. Since v_k appears in $\phi(X)$, potential $\phi(X)$ is relevant for message construction in SP.

($\Leftarrow$) Suppose SP determines potential $\phi(X)$ is relevant for message construction from a node N to a neighbour node sharing variables S. Then, $\phi(X)$ contains a non-evidence variable $v_1 \in S$ and another $v_k \notin S$. There are two cases to consider. Suppose $\phi(X) \in \mathcal{F}$. By definition, G must contain an edge (v_1, v_k), where G is the moralization of the domain graph of the factorization $\mathcal{F}$ of potentials at N. Therefore, LP determines potential $\phi(X)$ is relevant for message construction. Otherwise, suppose $\phi(X) \notin \mathcal{F}$. Then, $\phi(X)$ was built by SUMOUT in the elimination of another non-evidence variable v_{k-1}. By SUMOUT, this involved at least two potentials: ϕ_k containing v_k and v_{k-1}, and ϕ_{k-1} containing v_{k-1} and v_1. Again, there are two cases to consider. Suppose $\phi_{k-1} \in \mathcal{F}$. From ϕ_{k-1} and ϕ_k, G must contain a path $(v_1, v_{k-1}), (v_{k-1}, v_k)$. Hence, LP determines potential $\phi(X)$ is relevant for message construction. Otherwise, suppose $\phi_{k-1} \notin \mathcal{F}$. Thus, ϕ_{k-1} was built by SUMOUT in the elimination of another non-evidence variable v_{k-2}. By SUMOUT, this involved at least two potentials: ϕ_{k-1} containing v_{k-1} and v_{k-2}, and ϕ_{k-2} containing v_{k-2} and v_1. Once again, there are two cases to consider. Suppose $\phi_{k-2} \in \mathcal{F}$. From ϕ_{k-2}, ϕ_{k-1}, and ϕ_k, G

must contain a path (v_1, v_{k-2}), (v_{k-2}, v_{k-1}), (v_{k-1}, v_k). Thus, LP determines potential $\phi(X)$ is relevant. Now, suppose $\phi_{k-2} \notin \mathcal{F}$. Here, the above process can be repeated until SP eliminates its first non-evidence variable v_2 using only potentials in $\mathcal{F}$. The elimination of v_2 necessarily involves a potential ϕ_1 containing $v_1 \in S$ and $v_2 \notin S$. Therefore, by potentials $\phi_1, \ldots, \phi_{k-1}, \phi_k \in \mathcal{F}$, G must contain a path $(v_1, v_2), \ldots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k)$ from $v_1 \in S$ to $v_k \in X$ not separated by evidence E. Thus, LP determines potential $\phi(X)$ is relevant for message construction. ■

Even though SP and LP may eliminate variables in a different order, the relevant potentials in SP and LP are precisely the same.

Example 5 *In Example 4, LP determined at node N_2 that potentials $P(i)$, $P(l|h,i)$, $P(g|e)$, $P(e|d=0)$, $P(f|d=0)$, $P(h|f)$, $P(k|g,h)$, $P(j|g)$, $P(m|k)$, $P(n=1|l)$, $P(o|m)$, and $P(q|n=1)$ were relevant. In Example 2, SP determined at node N_2 that these same potentials were relevant.*

A deeper analysis of the proof of Theorem 1 reveals an intrinsic connection between paths in LP's domain graph and the potentials required during message construction in SP. SP can be seen as only building paths between X and S involving exclusively variables that were previously marginalized.

Example 6 *Recall SP in Example 2, where the separator $S = \{d, j, o, n, q\}$. The first variable to be eliminated is g. As shown in Figure 3 (i), there is a path (g, j) from g to S built using the potential $P(j|g)$ in $\mathcal{F}$. The second variable to be eliminated is h. As depicted in Figure 3 (ii), there is a path (h, g), (g, j) from h to S built using potentials $P(k|g,h)$ and $P(j|g)$ in $\mathcal{F}$. The third variable to be eliminated is f. Once again, as illustrated in Figure 3 (iii), there is a path from f to S built using potentials $P(h|f)$, $P(k|g,h)$, and $P(j|g)$ in $\mathcal{F}$. Paths from the remaining variables to be marginalized to S can be established similarly.*

5. Conclusion

The first main result of this paper is a comprehensive empirical evaluation of eight heuristics for determining elimination orderings in SP. Our second key contribution establishes that the relevant potentials in SP are exactly the relevant potentials in LP. This strengthens (Butz et al., 2016), where it was shown that SP is equivalent to LP, namely, the product of the messages are equal. Our experimental results suggest that SP does not require elimination orderings, provided that an optimal (or close to) join tree is built from the real-world BNs in Table 1. It is possible that elimination orderings are needed for larger BNs or when non-optimal join trees are used, since SP's performance degrades dramatically when applied on non-optimal join trees (Madsen et al., 2016). These studies remain as future work, as well the investigation on breaking ties when more than one variable is suitable for elimination.

References

C. J. Butz, J. S. Oliveira, A. E. dos Santos, and A. L. Madsen. Bayesian network inference with simple propagation. In *Proceedings of the Twenty-Ninth International FLAIRS Conference*, pages 650–655, 2016.

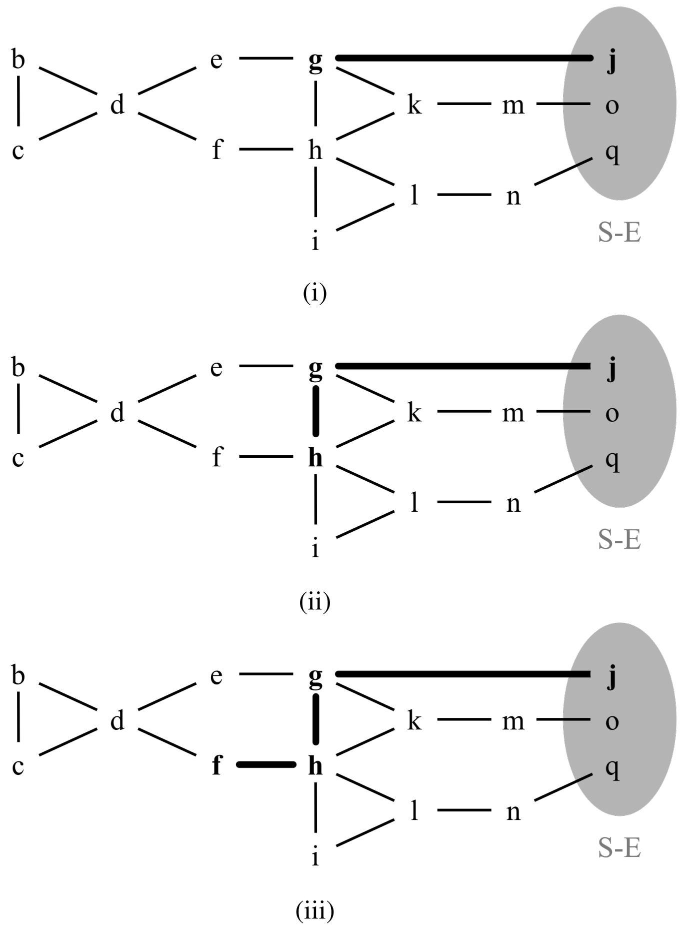

Figure 3: There exists a path in the moralization of the domain graph of $\mathcal{F}$ from each variable being marginalized to the separator $S = \{d, j, o, n, q\}$, traversing only previously marginalized variables: (i) (g, j); (ii) $(h, g), (g, j)$; and (iii) $(f, h), (h, g), (g, j)$.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

F. Jensen. *HUGIN API Reference Manual - V. 8.1*, 2014. URL `www.hugin.com`.

U. Kjærulff. Triangulation of graphs - algorithms giving small total state space. Technical Report R90-09, Aalborg University, Denmark, March 1990.

U. B. Kjærulff and A. L. Madsen. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer, 2nd edition, 2013.

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

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

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

A. L. Madsen. Improvements to message computation in lazy propagation. *International Journal of Approximate Reasoning*, 51(5):499–514, 2010.

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

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

A. L. Madsen, C. J. Butz, J. S. Oliveira, and A. E. dos Santos. On tree structures used by simple propagation for bayesian networks inference. In *Proceedings of the Twenty-Ninth Canadian Artificial Intelligence Conference*, pages 207–212, 2016.

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

G. Shafer. *Probabilistic Expert Systems*, volume 67. Philadelphia: Society for Industrial and Applied Mathematics, 1996.