On the Tree Structure Used by Lazy Propagation for Inference in Bayesian Networks

Anders L. Madsen[1,2] and Cory Butz[3]

[1] HUGIN EXPERT A/S, Aalborg, Denmark
anders@hugin.com
[2] Department of Computer Science, Aalborg University, Denmark
[3] Department of Computer Science, University of Regina, Canada
butz@cs.uregina.ca

Abstract. Lazy Propagation (LP) is a propagation scheme for belief update in Bayesian networks based upon Shenoy-Shafer propagation. So far the secondary computational structure has been a junction tree (or strong junction tree). This paper describes and shows how different tree structures can be used for LP. This includes the use of different junction trees and the maximal prime subgraph decomposition organised as a tree. The paper reports on the results of an empirical evaluation on a set of real-world Bayesian networks of the performance impact of using different tree structures in LP. The results indicate that the tree structure can have a significant impact on both time and space performance of belief update.

Keywords: Bayesian networks, inference, tree structure.

1 Introduction

A Bayesian network (BN) is an efficient knowledge base for representing uncertain knowledge [22, 3, 8, 9]. It consists of a graph specifying dependence and independence relations over a set of variables and a set of conditional probability distributions (CPDs) encoding the strengths of the dependence relations effectively combining elements of probability and graph theory. Due to the intuitive graphical nature of BNs, they have and are being used for handling uncertainty in a wide range of domains.

The key element in handling uncertainty with BNs is to perform probabilistic inference or belief update, i.e., to compute posterior probabilities given (partial or incomplete) information about the state of the domain. As both exact and approximate probabilistic inference in BNs are NP-hard [2, 4], methods that in the worst case have exponential complexity are justified (unless P=NP). Methods such as Variable Elimination (VE) [29] (equivalent to Fusion [27] and Bucket elimination [5]), Symbolic Probabilistic Inference (SPI) [24, 11] and Arc-Reversal (AR) [20, 25] are often referred to as *direct methods* as they focus on computing a single posterior marginal by manipulating the set of CPDs directly. On the other hand, methods such as Lauritzen-Spiegelhalter propagation [10], HUGIN

L.C. van der Gaag (Ed.): ECSQARU 2013, LNAI 7958, pp. 400–411, 2013.

propagation [7] and Shenoy-Shafer propagation [28] are referred to as *indirect methods* as they focus on computing all posterior marginals by passing messages in a secondary computational structure.

A number of hybrid algorithms combining direct and indirect methods have been proposed such as, for instance, *factor trees* [1] and LP [18]. LP is based on a Shenoy-Shafer propagation scheme using a direct method for message computation [12–14]. In [19] an algorithm for decomposing a BN into its maximal prime subgraphs is presented. The work reported in this paper was motivated by the potential use of the maximal prime subgraph decomposition (MPD) organised into a tree as a computational structure of LP. We evaluate the use of different tree structures in LP. This includes evaluating the potential use of the MPD organised into a tree, using a (near-) optimal junction tree versus a non-optimal junction tree and a junction tree with a single node as the tree structure. The results of an extensive empirical evaluation indicate that the tree structure can have a significant impact on both time and space performance of belief update.

The rest of the paper is organised as follows. Section 2 contains preliminaries. Section 3 presents the VE and LP algorithms as used in this paper and Section 4 describes the use of LP on different tree structures. Section 5 describes the design of the empirical evaluation and the results. Section 6 discusses the findings presented in this paper. Our conclusions are contained in Section 7.

2 Preliminaries

A (discrete) *BN* $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ consists of a set of random variables $\mathcal{X}$, an acyclic, directed graph (DAG) $G = (V, E)$ where $V \sim \mathcal{X}$ is the set of vertices and E is the set of edges and a set of CPDs $\mathcal{P}$. It represents a factorization of a joint probability distribution into a set of conditionals:

$$P(\mathcal{X}) = \prod_{X \in \mathcal{X}} P(X \,|\, \mathrm{pa}(X)), \tag{1}$$

where $\mathrm{pa}(X)$ denotes the parents of X in G and $\mathrm{fa}(X) = \mathrm{pa}(X) \cup \{X\}$.

Belief update is defined as the task of computing the posterior marginal distribution $P(X \,|\, \epsilon)$ for each non-observed variable $X \in \mathcal{X}$ given a set of evidence ϵ. An *evidence function* $f(X)$ is used to force an evidence variable X to its observed state x by assigning the value 1 to x and 0 otherwise. The set of observed variables is denoted $\mathcal{X}_\epsilon$. Barren variables are variables that are neither evidence nor target variables and have only barren descendants, if any [25].

A *probability potential* on *domain* $\mathrm{dom}(\phi) = \mathcal{Y}$ is a function ϕ such that $\phi(y) \geq 0$, for each configuration $y \in \mathcal{Y}$ and at least one $\phi(y) > 0$ [26]. A *conditional probability potential* ϕ of H given T is a probability potential of H when T is known where $\mathrm{dom}(\phi) = H \cup T$ is divided into head variables H denoted $\mathrm{head}(\phi)$ and tail variable T denoted $\mathrm{tail}(\phi)$. That is, $\mathrm{head}(\phi)$ and $\mathrm{tail}(\phi)$ are the conditioned and conditioning variables of $\mathrm{dom}(\phi)$, respectively

The *domain graph* representation $G(\phi) = (V, E)$ of a potential ϕ has vertices $V = \mathrm{dom}(\phi)$ and edges $E = \{(H_1, H_2), (H_2, H_1) \,|\, H_1, H_2 \in \mathrm{head}(\phi)\} \cup \{(T, H) \,|$

$H \in \text{head}(\phi), T \in \text{tail}(\phi)\}$. The notion of barren variables can be extended to domain graphs [13].

Let G be an undirected graph. A *clique* C is a *maximal, complete subgraph* of G. If the vertices V of a undirected graph G can be partitioned into a triple (V', S, V'') of nonempty sets where S is a complete separator of V' and V'' in G such that every path from a vertex in V' to a vertex in V'' includes a vertex in S, then G is *decomposable*; otherwise G is *prime*. A subgraph $G(U)$ of a graph $G = (V, E)$ is a *maximal prime subgraph* of G, if $G(U)$ is prime and $G(W)$ is decomposable for all W with $U \subset W \subseteq V$ [19]. The set of maximal prime subgraphs of a Bayesian network $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ are defined with respect to G^M.

A junction tree representation $T = (\mathcal{C}, \mathcal{S})$ of $\mathcal{N}$ with cliques $\mathcal{C}$ and separators $\mathcal{S}$ is constructed from a triangulated graph G^T produced by triangulating the moral graph G^M of G. The size $s(C)$ of a clique (separator) $C \in \mathcal{C}$ ($S \in \mathcal{S}$) is defined as the combined state space size of C (S), i.e., $s(C) = \prod_{X \in C} ||X||$. The size of a junction tree T is defined as $s(T) = \sum_{C \in \mathcal{C}} s(C)$ and T over $\mathcal{N}$ is *optimal* if $s(T) \leq s(T')$ for any T' over $\mathcal{N}$. We denote an optimal junction as $\hat{T}$. The number of cliques in $\mathcal{C}$ is denoted $|\mathcal{C}|$. A junction tree with $|\mathcal{C}| = 1$ is denoted T_1.

The algorithm of [19] produces a cluster tree from a junction tree T by recursively aggregating cliques connected by incomplete separators (in G^M) to larger clusters where T should be minimal. The resulting cluster tree is referred to as the *MPD tree* $T' = (\mathcal{C}', \mathcal{S}')$ with clusters $\mathcal{C}'$ and (complete) separators $\mathcal{S}'$.

A junction tree $T = (\mathcal{C}, \mathcal{S})$ is initialised by associating each CPD $P \in \mathcal{P}$ with the smallest clique $A \in \mathcal{C}$ such that $\text{dom}(P) \subseteq A$. The set of CPDs associated with a cluster C' is defined by the aggregated cliques producing it.

For example, consider Asia [10] with BN $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$. Figure 1 shows G (i), G^M (ii), an optimal junction tree (not showing separators) $\hat{T} = (\hat{\mathcal{C}}, \hat{\mathcal{S}})$ with $|\hat{\mathcal{C}}| = 6$, $s(\hat{T}) = 40$ and $\max s(C) = 8$ (iii) and the MPD tree $T' = (\mathcal{C}', \mathcal{S}')$ with $|\mathcal{C}'| = 5$, $s(T') = 40$ and $\max s(C') = 16$ (iv). Each $P \in \mathcal{P}$ is associated with a clique $C \in \mathcal{C}$ that can hold it and BEL is the only clique with no P associated.

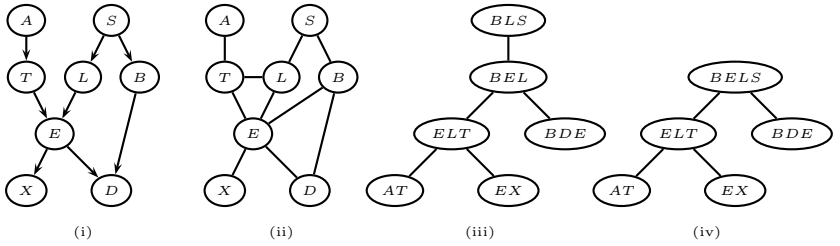

Fig. 1. (i) Asia (ii) G^M (iii) an optimal junction tree (iv) the MPD

In Asia only two cliques (BLS and BEL) are aggregated to form a single cluster ($BELS$) in $\mathcal{C}'$. The other cliques $\hat{\mathcal{C}} \setminus \{BLS, BEL\}$ remain clusters in $\mathcal{C}'$ as their adjacent separators are complete in G^M.

3 Belief Update

There exists a number of different approaches to exploiting the decomposition of $P(\mathcal{X})$ in (1) to perform belief update (efficiently). As mentioned above, this paper considers VE and LP.

3.1 Variable Elimination

In VE, a posterior marginal probability distribution $P(X \,|\, \epsilon)$ for a non-observed variable X is, in principle, computed by normalising:

$$P(X, \epsilon) = \sum_{Y \neq X} \prod_{P \in \mathcal{P}} P \prod_{Z \in \mathcal{X}_\epsilon} f(Z), \tag{2}$$

where $f(Z)$ is an evidence potential reflecting the instantiation of Z.

Barren variables are removed before variable elimination is performed. *Barren variables* have the property that when eliminated they produce a uniform likelihood over the conditioning variables and can therefore be eliminated without performing any computations. Also, we will assume that in (2) distributions are instantiated to reflect the evidence ϵ (as opposed to summing out $\mathcal{X}_\epsilon$).

The order $\rho = (Y_1, \ldots, Y_{|\mathcal{X} \setminus \{X\}|})$ in which the variable eliminations are performed is the *elimination order*. The elimination order can be identified using a range of different algorithms. Since optimal triangulation is NP-hard, heuristics are often used. The *fill-in-weight* (*fiw*) heuristic [6], for instance, aims to minimise the sum of the weights of the fill-in edges produced by a node elimination operation, i.e., $s_{fiw}(X) = \sum_{(Y_i, Y_j) \in F} ||Y_i|| \cdot ||Y_j||$, where F is the set of fill-ins added by the elimination of X.

3.2 Lazy Propagation

LP [12–14] is based on a Shenoy-Shafer scheme where messages are passed in two phases over a junction tree representation T of the BN $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ to propagate the evidence ϵ. After initialisation and prior to message passing, each $P \in \mathcal{P}$ such that $\mathrm{dom}(\phi) \cap \mathcal{X}_\epsilon \neq \emptyset$ is instantiated to reflect ϵ. Each clique C holds an initial clique potential $\Phi_C = \{P_{i_1}, \ldots, P_{i_n}\}$ which is a set of instantiated CPDs. Propagation of evidence is the process of *collecting* and *distributing* messages to and from a chosen root of T. When VE is used for message (marginal) computation, the message passed from clique A to clique B is computed as

$$\Phi_{A \to B} = \sum_{A \setminus B} (\Phi_A \cup \bigcup_{C \in \mathrm{adj}(A) \setminus \{B\}} \Phi_{C \to A}), \tag{3}$$

where $\mathrm{adj}(A)$ are the cliques adjacent to A in T. Prior to computing (3) barren variables and potentials corresponding to domain graphs over variables all separated from B given ϵ are removed. Notice that the result is a set of potentials. Notice also that the moralization step of the junction tree compilation in effect

ignores a lot of the information contained in a DAG. A key element in LP is to use information from the DAG to improve efficiency of inference. After message passing has terminated, $P(X \mid \epsilon)$ can be computed from any $C \in \mathcal{C}$ or $S \in \mathcal{S}$ such that $X \in C$ or $X \in S$. This version of LP is referred to as LPVE.

4 Tree Structure

In previous work on LP, the secondary computational tree structure of LP has been a junction tree (or in some cases a strong junction tree [17, 13]). In [19], the authors suggest using the MPD of $\mathcal{N}$ as the computational tree structure of LP. This paper evaluates the impact on performance of using different tree structures in LP. This includes different junction trees and MPD trees. The tree structure of LP is, in principle, a structure for caching intermediate results.

4.1 Junction Tree

There exists a number of different heuristics for generating a junction tree representation T of $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$. Some algorithms such as, for instance, *fiw* are based on node elimination where the next node elimination operation is based on the node with lowest scores, some are based on decomposing the graph G into its minimal separators and others are based on exhaustive search [21].

A special case is when the junction tree has only a single clique. LP over a single clique is, in principle, equivalent to VE. In a junction tree $T_1 = (\{\mathcal{X}\}, \emptyset)$ there is no caching of intermediate results. Each marginal $P(X \mid \epsilon)$ is computed from $\mathcal{P}$ given ϵ after removing barren variables and potentials corresponding to variables separated from X given ϵ. The computations corresponds to (2).

Proposition 1. *Let $A = \mathcal{X}$ be the single cluster in a junction tree $T = (\mathcal{C} = \{\mathcal{X}\}, \emptyset)$. We have*

$$P(A, \epsilon) = \prod_{\phi \in \Phi_A} \phi \prod_{i=1}^{n} f_i, \tag{4}$$

where $\Phi_A = \mathcal{P}$ is the set of potentials associated with A.

Proof. (4) is (1) now with evidence functions. □

4.2 Maximal Prime Subgraph Decomposition Tree

The nodes of the MPD tree T' represent maximal prime subgraphs in G^M whereas the nodes of a junction tree represent maximal complete subgraphs in G^T. As mentioned in Section 2, a MPD tree T' can be constructed from a junction tree T representing any minimal triangulation G^T of G by iteratively aggregating adjacent cliques connected by an incomplete separator in G^M. By construction the structure of T' is equivalent to T up to complete separators in G^M. Each cluster $C' \in \mathcal{C}'$ represents a connected set of cliques in $\mathcal{C}$.

Propagation of evidence in a MPD tree T' is similar to propagation of evidence in a junction tree as described in Section 3.2. Messages are *collected* to and *distributed* from a chosen root of T' where messages are computed as in (3).

Proposition 2. *Let A be a cluster in a MPD tree, let S be a neighboring separator and let $\epsilon = \{\epsilon_1, \ldots, \epsilon_n\}$ be the evidence. After a full round of message passing, we have*

$$P(A, \epsilon) = \prod_{\phi \in \Phi_A} \phi \prod_{i=1}^{n} f_i \prod_{C \in \mathrm{adj}(A)} \prod_{\phi' \in \Phi_{C \to A}} \phi',$$

$$P(S, \epsilon) = \prod_{\phi \in \Phi_{S \to A}} \phi \prod_{\phi' \in \Phi_{S \leftarrow A}} \phi',$$

where Φ_A is the set of potentials associated with A, $\Phi_{C \to A}$ is the set of potentials passed to A, and $\Phi_{S \to A}$ and $\Phi_{S \leftarrow A}$ are the sets of potentials passed over S.

Proof. The MPD tree T' corresponds to a triangulated graph G^T of G^M, where each maximal prime subgraph is made complete and its set of nodes is equivalent to the cliques of the junction tree created from G^T. □

The maximal prime subgraph can be independently triangulated to produce an optimal triangulation if each maximal prime subgraph is optimally triangulated. This does, however, not take into account the independence and barren variable properties induced by a specific set of evidence. Hence, an optimal junction tree may not be the best tree structure for belief update using LP given a specific set of evidence.

4.3 Example

In Asia of Figure 1 (i), consider the calculation of $P(D)$ and $P(E)$ using three different structures, namely, a single cluster tree, an optimal tree $\hat{T}$ in Figure 1 (iii), and the MPD tree in Figure 1 (iv). Using a single cluster with all variables to guide the computation will identify X as barren relative to $P(D)$ and eliminate the remaining variables, for example, in the order $\rho_D = (A, T, L, S, E, B)$ and identify $\{D, B, X\}$ as barren relative to $P(E)$ and eliminate the remaining variables, for instance, in the order $\rho_E = (A, T, L, S)$. Notice the amount of repeated computations. In an optimal tree $\hat{T}$, $P(E)$ and $P(D)$ can be computed from clique BDE after a collect to it (even though this may not be the optimal choice as $P(E)$ can be computed more efficiently from EX). Variables A and X are identified as barren relative to the message passed to BDE while B is not identified as barren as the elimination of S creates the potential $\phi(B, L)$. Note that the elimination of A and T is *cached* at BEL eliminating the repeated computations. Lastly, in the MPD tree T', the situation is the same as for $\hat{T}$ except that in the cluster $BELS$ there are more degrees of freedom to determine ρ than in the clique BEL as the latter case corresponds to restricting ρ (in $BELS$) to have S as the first variable. This may in some cases be suboptimal.

5 Experimental Analysis

We give an empirical evaluation of the performance impact of using different secondary computational structures in LP based upon a set of real-world BNs.

5.1 Setup

Table 1[1] shows statistics on the BNs used in the evaluation and their optimal (or believed to be near-optimal) junction tree ($\hat{T}$), a junction tree generated using s_{fiw} (T_{fiw}) and the MPD tree (T'), respectively. In the table $|\mathcal{Y}|$ is the cardinality of $\mathcal{Y}$ and sizes are on a log-scale in base 10. The junction trees have been generated using the *total weight* and *fill-in-weight* heuristics as implemented in the HUGIN tool [6, 16]. The test set consists of networks of different size and complexity in terms of the size of the tree structure.

For each network, one hundred sets of evidence have been generated at random. For each evidence set, LPVE computes the posterior marginal distribution of each non-evidence variable. The same set of evidence sets is used to evaluate each tree structure for a specific network. In the experiments, the *fiw* heuristic is applied to determine the online elimination order when computing messages and posterior marginals [15].

The experiments were performed using a Java implementation (Java (TM) SE Runtime Environment, Standard Edition (build 1.7.0_10-b18)) running on a Linux Ubuntu 12.10 (kernel 3.5.0-21-generic) PC with an Intel Core i7(TM) 920 Processor (2.67GHz) and 12 GB RAM.

5.2 Results

Table 2 presents the time performance results of the evaluation for $\hat{T}$, T_{fiw}, T_1 and T', respectively. Table 2 shows the sample average run-time in seconds and the sample variance for propagating one hundred sets of evidence generated at random, for each network and each type of secondary computational structure.

Table 3 shows size of the largest potential created during belief update using tree structures $\hat{T}$, T_{fiw}, T_1 and T', respectively. The table shows the sample average and variance when propagating one hundred sets of randomly generated evidence, for each network and each type of secondary computational structure. The time performance measurements include time for finding the on-line triangulation orders and do not include time used to generate the secondary computational structure. It is expected that on-line triangulation is more expensive for T_1 and T' than for $\hat{T}$ and T_{fiw}.

Notice that two different implementations of the *fill-in-weight* heuristic have been used. The junction trees have been generated using the HUGIN tool while the online triangulation have been generated using our own implementation. This may in part explain why T_1 produces a larger average largest potential size than the largest clique in T_{fiw}. The total cost of online triangulation is expected

[1] The size of the largest cluster for Diabetes cannot be represented using a Java double.

Table 1. Description of test BNs, $\hat{T}$, T_{fiw} and T' where * means that the triangulation is optimal, ** means that the triangulation has been created using a maximum of 200,000 separators and no * means that the best known triangulation is used

$\mathcal{N}$	$\lvert\mathcal{X}\rvert$	$\lvert\hat{\mathcal{C}}\rvert$	$\lvert\mathcal{C}_{fiw}\rvert$	$\lvert\mathcal{C}'\rvert$	max $s(\hat{C})$	max $s(C_{fiw})$	max $s(C')$	$s(\hat{T})$	$s(T_{fiw})$	$s(T')$
3nt*	58	41	41	22	3.5	3.7	16.8	4.1	4.4	16.8
Barley*	48	36	36	14	6.9	6.9	29.5	7.2	7.3	29.5
Diabetes	413	337	337	77	4.9	5.5	-	7.0	7.1	-
Hepar_II*	70	58	58	55	2.6	2.6	2.9	3.4	3.4	3.5
KK*	50	38	38	15	6.8	6.8	30.2	7.1	7.2	30.2
Mildew*	35	29	28	15	6.1	6.6	20.6	6.5	7.0	20.6
Munin1	189	162	160	70	7.6	7.9	69.2	7.9	8.3	69.2
Munin2	1,003	854	860	48	5.2	5.7	189.6	6.3	6.7	189.6
Munin3	1,044	904	904	53	5.2	5.2	174.5	6.5	6.5	174.5
Munin4	1,041	877	875	49	5.7	5.9	221.9	6.9	7.1	221.9
Water*	32	21	19	9	5.8	6.2	13.3	6.5	6.6	13.3
andes**	223	180	175	79	4.8	5.4	40.0	5.3	5.6	40.0
cc145*	145	140	140	13	3.0	3.0	3.0	3.6	3.6	3.6
cc245*	245	235	235	23	5.4	5.4	6.0	5.8	5.8	6.3
hailfinder*	56	43	43	29	3.5	3.5	11.6	4.0	4.0	11.6
medianus*	56	44	44	15	5.7	5.7	28.4	6.1	6.2	28.4
oow*	33	22	22	6	6.3	6.8	21.7	6.8	7.3	21.7
oow_bas*	33	19	19	8	5.7	6.2	18.4	6.3	6.6	18.4
oow_solo*	40	29	28	9	6.2	7.2	24.2	6.7	7.5	6.3
pathfinder*	109	91	91	86	4.5	4.5	6.8	5.3	5.3	6.8
sacso**	2,371	1,229	1,175	98	5.2	6.4	107.5	6.0	6.8	107.5
ship*	50	35	35	10	6.6	8.1	35.6	7.4	8.4	35.6
system_v57*	85	75	72	26	4.8	6.7	57.9	6.1	6.8	57.9
win95pts*	76	50	50	33	2.7	2.7	9.3	3.4	3.4	9.3

to be higher for T_1 as elimination orders on average are expected to be *longer* for this structure in the following sense. For T_1 all variables are in a single clique. This means that to compute any posterior marginal $P(X \mid \epsilon)$ all variables $\mathcal{X} \setminus \{X\}$ have to be eliminated (in principle) and the elimination order has length $|\mathcal{X}| - 1$. Using T_{fiw}, on the other hand, each marginal $P(X \mid \epsilon)$ is computed from any clique or separator containing X. Since the number of variables in the largest clique is usually much smaller than $|\mathcal{X}|$, the elimination orders are usually much shorter for T_{fiw}. The implementation of the online triangulation has not been optimised to cope with large domain graphs.

Observe that for a few networks average time performance on T_1 and T' is much worst than T_{fiw} and $\hat{T}$ with a high variance. For a few evidence sets the time performance is significantly worse for these structures. For instance, the average run-time performance on Diabetes is high with a high variance.

T_1 seems to have the worst time performance except for a few instances, while time performance of T' in one case is much worse than T_1 (as well as T_{fiw} and $\hat{T}$) and in a number of cases is comparable with the performance of T_{fiw} and $\hat{T}$. In almost all cases time performance of T_{fiw} is similar to the performance of $\hat{T}$,

Table 2. Run-time in seconds (mean ± standard deviation)

$\mathcal{N}$	$\hat{T}$	T_{fiw}	T_1	T'
3nt*	0.03±0.00	0.03±0.00	0.05±0.04	0.04±0.00
Barley*	0.13±0.18	0.15±0.21	0.34±0.64	0.33±0.61
Diabetes	0.45±0.39	0.47±0.41	27.69±72.84	93.83±282.90
Hepar_II*	0.05±0.00	0.05±0.03	0.1±0.07	0.05±0.00
KK*	0.12±0.15	0.14±0.18	0.38±0.63	0.22±0.32
Mildew*	0.06±0.06	0.08±0.10	0.22±0.67	0.17±0.50
Munin1	0.84±1.99	1.49±4.40	4.49±19.47	3.98±18.53
Munin2	0.6±0.25	0.61±0.26	8.37±11.50	1.6±1.90
Munin3	0.81±0.41	0.81±0.41	25.14±46.30	8.06±17.20
Munin4	0.75±0.38	0.76±0.41	15.25±22.70	3.96±6.13
Water*	0.08±0.08	0.07±0.06	0.09±0.11	0.09±0.11
andes**	0.19±0.09	0.18±0.09	0.69±0.69	0.51±0.39
cc145*	0.12±0.06	0.12±0.06	0.14±0.10	0.11±0.06
cc245*	0.27±0.12	0.26±0.12	0.35±0.30	0.26±0.12
hailfinder*	0.04±0.00	0.04±0.00	0.09±0.07	0.04±0.00
medianus*	0.05±0.03	0.06±0.04	0.1±0.14	0.09±0.13
oow*	0.1±0.12	0.14±0.22	0.17±0.44	0.13±0.24
oow_bas*	0.05±0.04	0.07±0.08	0.08±0.10	0.06±0.06
oow_solo*	0.1±0.12	0.29±0.61	0.79±3.29	0.55±2.14
pathfinder*	0.15±0.11	0.15±0.11	0.15±0.13	0.14±0.11
sacso**	0.66±0.25	0.67±0.26	44.51±76.62	1.85±2.20
ship*	0.25±0.46	1.4 ±4.90	1.42±5.13	0.88±3.76
system_v57*	0.09±0.05	0.12±0.14	0.71±1.56	0.6±1.35
win95pts*	0.05±0.00	0.05±0.00	0.12±0.07	0.08±0.04

while $\hat{T}$ is better than T_{fiw} in a few cases. On the other hand, in many cases the space performance of T_1 and T' is better than the space performance of T_{fiw} and $\hat{T}$. There are a few significant exceptions though, which is surprising.

6 Discussion and Analysis

Traditionally, LP has been based on message passing in a junction tree representation of a BN. This paper has described and evaluated how different tree structures can be used for LP. This includes different junction trees, MPD trees and junction trees with a single clique.

The identification of the MPD tree can be relatively efficient compared to finding the optimal junction tree, which can be a relatively expensive operation. The MPD is identified using a minimal triangulation and a linear search guided by the junction tree. The classical triangulation algorithm LEX M [23] can be used to determine a minimal triangulation with time complexity $\mathcal{O}(ne)$, where n is the number of vertices and e is the number of edges in the graph [23]. The complexity of constructing the MPD tree from a minimal junction tree is $\mathcal{O}(n^2)$ [19]. In the evaluation, we have generated MPD trees from the junction trees generated using *total-weight*.

Table 3. Size of largest potential (mean ± standard deviation)

$\mathcal{N}$	$\hat{T}$	T_{fiw}	T_1	T'
3nt*	2.9±3.0	2.9±3.1	2.6±2.7	2.6±2.7
Barley*	5.7±6.2	5.7±6.2	5.2±5.6	5.3±5.7
Diabetes	4.6±4.5	5.0±5.1	6.3±7.0	7.2±7.9
Hepar_II*	2.0±2.1	2.0±2.1	2.0±2.1	2.0±2.1
KK*	5.7±6.1	5.7±6.1	5.3±5.7	5.3±5.7
Mildew*	5.3±5.6	5.6±6.0	5.4±5.9	5.4±5.9
Munin1	6.3±6.8	6.6±7.0	6.0±6.7	6.1±6.8
Munin2	4.3±4.5	4.6±5.0	3.6±3.9	3.6±3.9
Munin3	4.6±4.8	4.6±4.8	5.1±5.5	5.1±5.5
Munin4	5.0±5.2	5.2±5.4	4.9±5.3	4.9±5.3
Water*	4.9±5.2	4.9±5.2	4.6±5.0	4.6±5.0
andes**	3.5±3.9	3.5±3.9	2.8±3.1	2.8±3.1
cc145*	2.2±2.3	2.2±2.3	2.2±2.3	2.2±2.3
cc245*	4.0±4.2	4.0±4.2	3.9±4.2	3.9±4.2
hailfinder*	3.0±3.1	3.0±3.1	2.8±3.0	2.8±3.0
medianus*	4.6±5.0	4.7±5.1	4.4±5.3	4.4±5.3
oow*	5.4±5.7	5.9±6.3	5.8±6.5	5.5±6.2
oow_bas*	4.9±5.2	5.4±5.7	5.1±5.6	5.1±5.6
oow_solo*	5.5±5.7	6.1±6.5	5.8±6.3	5.9±6.6
pathfinder*	3.8±4.0	3.8±4.0	3.8±4.0	3.8±4.0
sacso**	3.8±4.1	4.3±4.7	3.4±3.9	3.4±3.8
ship*	5.9±6.1	6.9±7.5	6.5±7.3	5.7±6.2
system_v57*	4.5±4.4	5.5±6.0	5.8±6.4	5.9±6.5
win95pts*	2.1±2.2	2.1±2.2	1.9±2.0	1.9±2.0

A junction tree is a caching structure. It caches in the separator potentials the results of intermediate variable elimination operations. This may give $\hat{T}$ an advantage over T_1 which has to identify an complete elimination order for each posterior marginal. On the other hand, $\hat{T}$ is *wide enough* to accommodate any set of evidence. This may be a disadvantage compared to T_1, which can exploit all information in the structure of the evidence. The results reported in this paper indicates that for only a few networks the time performance is insensitive to the tree structure, e.g., for *pathfinder* and *Water* the four structures considered produce almost equal time performance. In some cases the time performance is almost the same for $\hat{T}$, T' and T_{fiw}. This is the case, e.g., for *Hepar_II* and *hailfinder*. In other cases, the time performances of $\hat{T}$ and T_{fiw} are similar, whereas the time performances of T' and T_1 are much worse. This is the case, e.g., for *Barley*, *Munin4* and *Mildew*. In some cases the time performance of T_1 or/and T' is poor compared to the other algorithms. In these cases, the time performance variance is very high. This indicates that the time performance is poor on a few sets of evidence producing a high average time performance. The poor time performance is due to large potentials created during belief update and the large potentials are created due to a poor elimination order. It should be noted that in some cases $\hat{T}$ is not known to be optimal (finding the optimal triangulation is infeasible as the number of minimum separators in G^M is large).

In general, the evaluation illustrates that the tree structure can have a significant impact on performance. In most cases, $\hat{T}$ and T_{fiw} produce the best results. In almost all cases (except one) T_1 produced the worst results. Notice that in some cases T_1 produces a larger largest potential than T_{fiw}.

7 Conclusion

This paper has considered the impact of the secondary computational structure used by LP in belief update. The results of the empirical evaluation indicate that the tree structure can have a significant impact on both time and space performance of belief update. The structures $\hat{T}$ and T_{fiw} most often produced the best performance on the networks considered in the evaluation.

Future work includes assessing the impact of using a binary tree structure such as the binary join tree [27] as well as evaluating different variants of LP such as LP using AR or SPI as the message computation algorithm. In addition, the option to consider *almost* complete separators as complete should be considered in order to divide large maximal prime subgraphs into smaller clusters, i.e., to increase the level of caching in the tree structure.

Acknowledgments. We would like to thank the reviewers for their insightful comments, which have improved the quality of the paper.

References

1. Bloemeke, M., Valtorta, M.: A Hybrid Algorithm to compute Marginal and Joint Beliefs in Bayesian Networks and Its Complexity. In: Proc. of the UAI, pp. 16–23 (1998)
2. Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence 42(2-3), 393–405 (1990)
3. Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J.: Probabilistic Networks and Expert Systems. Springer (1999)
4. Dagum, P., Luby, M.: Approximating probabilistic inference in Bayesian belief netwoks is NP-hard. Artificial Intelligence 60, 141–153 (1993)
5. Dechter, R.: Bucket elimination: A unifying framework for probabilistic inference. Artificial Intelligence 113(1-2), 41–85 (1999)
6. Jensen, F.V.: HUGIN API Reference Manual. HUGIN EXPERT A/S, Reference Manual for the HUGIN version 7.7 (2012), `http://www.hugin.com`
7. Jensen, F.V., Lauritzen, S.L., Olesen, K.G.: Bayesian updating in causal probabilistic networks by local computations. Computational Statistics Quarterly 4, 269–282 (1990)
8. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer (2007)
9. Kjærulff, U.B., Madsen, A.L.: Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis, 2nd edn. Springer (2012)
10. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, B 50(2), 157–224 (1988)

11. Li, Z., D'Ambrosio, B.: Efficient Inference in Bayes Networks as a Combina torial Optimization Problem. Int. J. of Approximate Reasoning 11(1), 55–81 (1994)
12. Madsen, A.L.: An empirical evaluation of possible variations of lazy propagation. In: Proc. of the UAI, pp. 366–373 (2004)
13. Madsen, A.L.: Variations Over the Message Computation Algorithm of Lazy Propagation. IEEE TSMC Part B 36(3), 636–648 (2006)
14. Madsen, A.L.: Improvements to Message Computation in Lazy Propagation. Int. J. of Approximate Reasoning 51(5), 499–514 (2010)
15. Madsen, A.L., Butz, C.J.: On the Importance of Elimination Heuristics in Lazy Propagation. In: Sixth European Workshop on Probabilistic Graphical Models, pp. 227–234 (2012)
16. Madsen, A.L., Jensen, F.V., Kjærulff, U.B., Lang, M.: Hugin - the tool for bayesian networks and influence diagrams. International Journal on Artificial Intelligence Tools 14(3), 507–543 (2005)
17. Madsen, A.L., Jensen, F.V.: Lazy Evaluation of Symmetric Bayesian Decision Problems. In: Proc. of the UAI, pp. 382–390 (1999)
18. Madsen, A.L., Jensen, F.V.: Lazy propagation: A junction tree inference algorithm based on lazy evaluation. Artificial Intelligence 113(1-2), 203–245 (1999)
19. Olesen, K.G., Madsen, A.L.: Maximal Prime Subgraph Decomposition of Bayesian Networks. IEEE TSMC Part B 32(1), 21–31 (2002)
20. Olmsted, S.M.: On representing and solving decision problems. PhD thesis, Department of Engineering-Economic Systems, Stanford University, CA (1983)
21. Ottosen, T.J., Vomlel, J.: All roads lead to Rome - New search methods for the optimal triangulation problem. Int. J. of Approximate Reasoning 53(9), 1350–1366 (2012)
22. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Series in Representation and Reasoning. Morgan Kaufmann Publishers, San Mateo (1988)
23. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM Journal of Computing 5(2), 266–283 (1976)
24. Shachter, R., D'Ambrosio, B., DelFavero, B.: Symbolic probabilistic inference in belief networks. In: Proc. Eighth National Conference on AI, pp. 126–131 (1990)
25. Shachter, R.D.: Evaluating influence diagrams. Operations Research 34(6), 871–882 (1986)
26. Shafer, G.R.: Probabilistic Expert Systems. SIAM (1996)
27. Shenoy, P.P.: Binary join trees for computing marginals in the Shenoy-Shafer architecture. Int. J. of Approximate Reasoning 17(2-3), 239–263 (1997)
28. Shenoy, P.P., Shafer, G.: Axioms for probability and belief-function propagation. In: Proc. of the UAI, pp. 169–198 (1990)
29. Zhang, N.L., Poole, D.: A simple approach to bayesian network computations. In: Proc. of the Canadian Conference on AI, pp. 171–178 (1994)