On Tree Structures Used by Simple Propagation

Anders L. Madsen[1,2(✉)], Cory J. Butz[3], Jhonatan S. Oliveira[3], and André E. dos Santos[3]

[1] Hugin Expert A/S, Aalborg, Denmark
anders@hugin.com
[2] Department of Computer Science, Aalborg University, Aalborg, Denmark
[3] Department of Computer Science, University of Regina, Regina, Canada

Abstract. *Simple Propagation* (SP) is a new junction tree-based algorithm for probabilistic inference in discrete Bayesian networks. It is similar to Lazy Propagation, but uses a simpler approach to exploit the factorization during message computation. The message construction is based on a *one-in, one-out*-principle meaning a potential has at least one non-evidence variable in the separator and at least one non-evidence variable not in the separator. This paper considers the use of different tree structures to guide the message passing in SP and reports on an experimental analysis using a set of real-world Bayesian networks.

Keywords: Bayesian networks · Inference · Simple propagation

1 Introduction

The Simple Propagation (SP) algorithm was introduced in [1] as a new algorithm for belief update in Bayesian networks. SP is similar to LP as it proceeds by message passing in a join tree or a junction tree taking advantage of a decomposition of clique and separator potentials, but it takes a simpler approach to the computation of potentials in the message passing phase of belief update. In SP, message construction is based on a *one-in, one-out*-principle meaning a potential has at least one non-evidence variable in the separator and at least one non-evidence variable not in the separator.

In this paper, we empirically analyze the role that the choice of the tree structure plays when SP conducts inference. We consider three kinds of tree structures, namely, optimal (or believed to be close to optimal) junction trees, junction trees produced by the *fill-in-weight* heuristic, and *maximal prime subgraph decomposition* (MPD) trees [4]. When using optimal junction trees or the ones produced by the fill-in-weight heuristic, experimental results shows that in fewer than half the cases, the space cost of SP is almost invariant. Moreover, the time cost of SP in these two categories is approximately the same in 14 out of 28 cases. On the other hand, SP often runs out of memory on MPD trees.

R. Khoury and C. Drummond (Eds.): Canadian AI 2016, LNAI 9673, pp. 207–212, 2016.
DOI: 10.1007/978-3-319-34111-8_26

2 Preliminaries and Notation

A Bayesian network $BN\ \mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ consists of a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$, where $\mathrm{dom}(X)$ is the state space of X and $||X|| = |\mathrm{dom}(X)|$, 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 $\mathcal{P}$ of conditional probability distributions (CPDs). A BN represents a decomposition of the joint probability distribution $P(\mathcal{X}) = \prod_{X \in \mathcal{X}} P(X \,|\, \mathrm{pa}(X))$, where $\mathrm{pa}(X)$ denotes the parents of X in G and $\mathrm{fa}(X) = \mathrm{pa}(X) \cup \{X\}$. For example, a BN $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ over six variables is shown by G in Fig. 1, where we assume $||X_1|| = ||X_4|| = 3$, $||X_2|| = ||X_5|| = ||X_6|| = 4$, and $||X_3|| = 2$.

Belief update is the task of computing the posterior marginal probability distribution $P(X \,|\, \epsilon)$ for each non-observed variable $X \in \mathcal{X} \setminus \mathcal{X}_\epsilon$ given evidence ϵ assumed to be instantiations of variables $\mathcal{X}(\epsilon)$. A variable X is a *barren* w.r.t. a set $T \subseteq \mathcal{X}$, evidence ϵ, and DAG G, if $X \notin T$, $X \notin \mathcal{X}_\epsilon$ and X only has barren descendants in G, if any [6]. The notion of barren variables can be extended to graphs with both directed and undirected edges [3].

A junction tree $T = (\mathcal{C}, \mathcal{S})$ of $\mathcal{N}$ is created by moralization and triangulation of G (see, e.g., [2]), where $\mathcal{C}$ denotes the cliques and $\mathcal{S}$ denotes the separators of T. The state space size of clique or separator A is defined as $s(A) = \prod_{X \in A} ||X||$.

3 Simple Propagation

SP [1] can be considered a simplification of LP [5] with respect to how messages are computed. The basic idea is to maintain a decomposition of clique and separator potentials and to exploit independence relations induced by evidence and barren variables during belief update. Each CPD $P \in \mathcal{P}$ is associated with a clique C s.t. $\mathrm{dom}(P) \subseteq C$, where P is reduced to reflect ϵ. Next messages are passed over the computational tree structure relative to a selected root of T.

In SP, the *one-in, one-out*-principle is applied when a clique sends a message to a neighbouring clique over a separator S. The *one-in, one-out*-principle states that a probability potential ϕ has at least one non-evidence variable in S and another variable X not in S and SP can eliminate X. The computation of messages in SP is performed using the Simple Message Computation (SMC) algorithm shown as Algorithm 1. The SumOut algorithm corresponds to Variable Elimination summing out X from the product of $\Phi_X = \{\phi \in \mathcal{F} | X \in \mathrm{dom}(\phi)\}$ and replacing Φ_X in $\mathcal{F}$ with the result.

Procedure *SMC*$(\mathcal{F}, S, \mathcal{X}(\epsilon))$
1 $\quad \mathcal{F} = \textsc{RemoveBarren}(\mathcal{F}, S)$
2 $\quad$ **while** $\exists \phi(\mathcal{Y}) \in \mathcal{F}$ *with* $X \notin (S \setminus \mathcal{X}(\epsilon))$ *and* $X' \in (S \setminus \mathcal{X}(\epsilon))$ **do**
3 $\quad\quad \mathcal{F} = \textsc{SumOut}(X, \mathcal{F})$
$\quad$ **end**
4 $\quad$ **return** $\{\phi(\mathcal{Y}) \in \mathcal{F} \mid \mathcal{Y} \subseteq S\}$

Algorithm 1. The Simple Message Computation algorithm.

In addition to passing the potentials created by Algorithm 1, SP also passes any potential ϕ for which $\text{dom}(\phi) \subseteq S$, i.e., the potentials where the domain is a subset of the separator. After a full round of message passing, the posterior marginal $P(X \mid \epsilon)$ can be computed from any clique or separator containing X.

4 Computational Tree Structure

The computational tree structure induces a partial order on the set of possible (implicit) elimination orders produced by the SMC algorithm during the message passing step of belief update. As SP maintains a decomposition of clique and separator potentials, it becomes sensitive to the order in which variables are eliminated during message passing. The number of variables to be eliminated when sending a message from A to B is determined by $A \setminus B$ which is influenced by $|A|$ and $|B|$. Thus, the impact of the (implicit) elimination order tends to increase with $|A \setminus B|$. As SP uses the same potentials for message computation as LP, we can rely on the correctness considerations of [4].

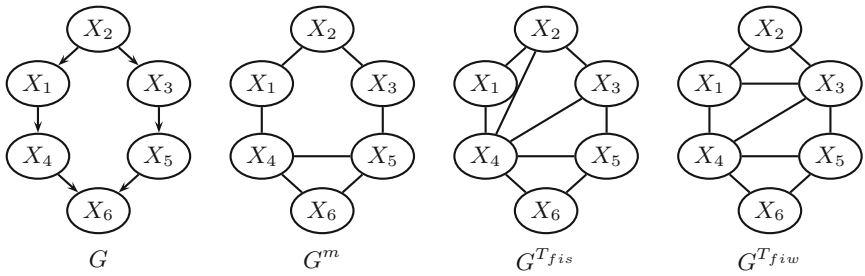

Fig. 1. A DAG G with G^m and triangulations using fill-in-size and fill-in-weight.

To illustrate the impact of using different computational tree structures, consider the moral graph G^m and two triangulations of G^m in Fig. 1. The fill-in-size heuristic may produce the elimination order $\sigma_{fis} = (X_6, X_1, X_5, X_2, X_3, X_4)$, while the order $\sigma_{fiw} = (X_6, X_5, X_2, X_1, X_3, X_4)$ will be produced by the fill-in-weight heuristic (optimal w.r.t. $s(T)$).

The junction trees T_{fis} and T_{fiw} produced are shown in Fig. 2. This figure also shows the MPD tree T_{MPD} for $\mathcal{N}$ constructed from T_{fiw} by merging cliques connected by incomplete separators in G^m. For simplicity, we assume that clique $X_4X_5X_6$ is selected as root of each tree.

Due to space limitations, we only consider the first message passed in each of the computational trees shown in Fig. 2. For T_{fis}, the first message is from clique $X_1X_2X_4$ to $X_2X_3X_4$. The clique potential $\pi_{X_1X_2X_4} = (\{P(X_2), P(X_1 \mid X_2), P(X_4 \mid X_1)\})$ has two CPDs satisfying the *one-in, one-out*-principle: $P(X_1 \mid X_2)$ and $P(X_4 \mid X_1)$. Notice $P(X_2)$ does not satisfy the principle as all domain variables are in S. Both $P(X_1 \mid X_2)$ and $P(X_4 \mid X_1)$ satisfy the condition in line 2

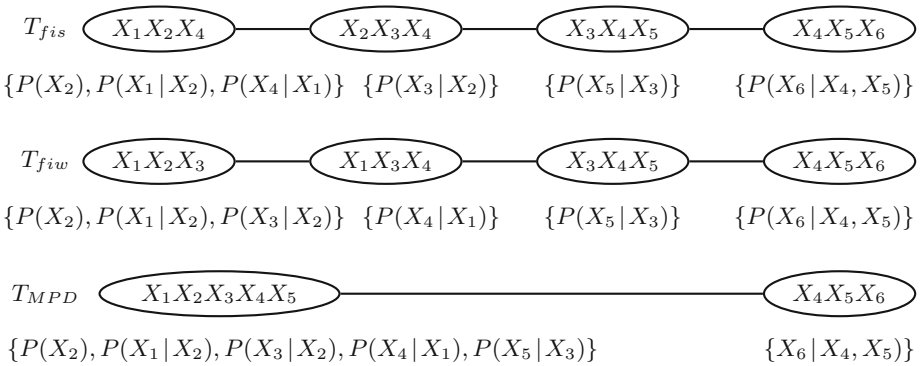

Fig. 2. Three different computational tree structures.

of the SMC algorithm and variable X_1 can be eliminated using the SumOut-algorithm producing $\phi(X_4 \mid X_2)$. The message is then

$$\pi_{X_1X_2X_4 \to X_2X_3X_4} = (\{P(X_2), \phi(X_4 \mid X_2)\}). \tag{1}$$

For T_{fiw}, the first message is from clique $X_1X_2X_3$ to $X_1X_3X_4$. The potential $\pi_{X_1X_2X_3} = (\{P(X_2), P(X_1 \mid X_2), P(X_3 \mid X_2)\})$ has two CPDs satisfying the *one-in, one-out*-principle: $P(X_1 \mid X_2)$ and $P(X_3 \mid X_2)$. Notice $P(X_2)$ does not satisfy the principle. Both $P(X_1 \mid X_2)$ and $P(X_3 \mid X_2)$ satisfy the condition in line 2 of the SMC algorithm and variable X_2 can be eliminated using the SumOut-algorithm producing $\phi(X_1, X_3)$. The message is then

$$\pi_{X_1X_2X_3 \to X_1X_3X_4} = (\{\phi(X_1, X_3)\}). \tag{2}$$

For T_{MPD}, the first message is from clique $X_1X_2X_3X_4X_5$ to $X_4X_5X_6$. Potential $\pi_{X_1X_2X_3X_4X_5} = (\{P(X_2), P(X_1 \mid X_2), P(X_3 \mid X_2), P(X_4 \mid X_1), P(X_5 \mid X_3)\})$ has two CPDs satisfying the *one-in, one-out*-principle: $P(X_4 \mid X_1)$ and $P(X_5 \mid X_3)$. Assume variable X_1 selected for elimination first producing the updated set $\mathcal{F} = \{P(X_2), P(X_3 \mid X_2), P(X_5 \mid X_3), \phi(X_4 \mid X_2)\}$. Next, $P(X_5 \mid X_3)$ and $\phi(X_4 \mid X_2)$ satisfy the condition in line 2. Assume variable X_2 is selected for elimination producing the updated set $\mathcal{F} = \{P(X_5 \mid X_3), \phi(X_4, X_3)\}$. Next, $P(X_5 \mid X_3)$ and $\phi(X_3, X_4)$ satisfy the condition in line 2. Variable X_3 is the last variable eliminated producing the updated set $\mathcal{F} = \{\phi(X_4, X_5)\}$. The message is then

$$\pi_{X_1X_2X_3X_4X_5 \to X_4X_5X_6} = (\{\phi(X_4, X_5)\}). \tag{3}$$

The important point to notice is that different tree structures lead to different messages being constructed by SP, as shown in Eqs. (1), (2), and (3). The next section reports on an empirical evaluation of the performance impact of different tree structures.

5 Experimental Analysis

Table 1 shows statistics on a sample of 10 out of 28 BNs used in the experiments and information on the junction tree $\hat{T}$, the junction tree T_{fiw} and the MPD tree T_{MPD}, where sizes are on a log-scale in base 10. Table 2 shows the average largest factor over 100 sets of random evidence created by SP during belief update (this includes computing marginals using Variable Elimination). The empty entries denote examples where SP ran out of memory on some evidence sets.

In fewer than half the cases, the space cost of SP is almost invariant to the use of $\hat{T}$ or T_{fiw} as the computational structure. These are the cases where $s(\hat{T}) \approx s(T_{fiw})$. In the cases where $s(\hat{T}) << s(T_{fiw})$, there is a large difference in the average size of the largest factor created by SP. SP is not able to perform belief update in 20 out of 28 networks using T_{MPD} as the computational structure.

Table 1. Information on 10 out of the 28 Bayesian networks and the computational tree structures used in the experiments.

					max	max	max			
$\mathcal{N}$	$\lvert\mathcal{X}\rvert$	$\lvert\hat{\mathcal{C}}\rvert$	$\lvert\mathcal{C}_{fiw}\rvert$	$\lvert\mathcal{C}'\rvert$	$s(\hat{C})$	$s(C_{fiw})$	$s(C')$	$s(\hat{T})$	$s(T_{fiw})$	$s(T')$
ADAPT_DX09_T2	671	489	489	284	3.1	3.5	29.6	4	5	30
Amirali_network	681	556	555	461	6.9	7.5	41.5	7	8	42
Heizung.	44	28	28	14	7.6	7.6	29.2	8	8	29
Hepar_II	70	58	58	55	2.6	2.6	2.9	3	3	4
Mildew	35	29	28	15	6.1	6.6	20.6	7	7	21
Munin1	189	162	160	70	7.6	7.9	69.2	8	8	69
andes	223	180	175	79	4.8	5.4	40.0	5	6	40
cc245	245	235	235	232	5.4	5.4	6.0	6	6	6
sacso	2371	1229	1175	980	5.2	6.4	107.5	6	7	107
ship	50	35	35	10	6.6	8.1	35.6	7	8	36

Table 2. Average space cost of belief update using SP.

$\mathcal{N}$	$\mu(\hat{T})$	$\sigma(\hat{T})$	$\mu(T_{fiw})$	$\sigma(T_{fiw})$	$\mu(T_{MPD})$	$\sigma(T_{MPD})$
ADAPT_DX09_T2	520.09	315372	727.13	845205		
Amirali_network	70716.8	3.6E+10	1093871.04	1.3E+13		
Heizung.	4091131	1.3E+14	3542217.5	1.2E+14		
Hepar_II	106.5	15384.4	106.5	15384.4	186.58	71361
Mildew	219317	1.6E+11	580396.08	1.7E+12		
Munin1	2499265	6.2E+13	4158649.69	1.3E+14		
andes	7958.68	3.1E+08	13922.2	1.4E+09		
cc245	21764.6	3.3E+09	21764.6	3.3E+09	37384.3	1.4E+10
sacso	12887.4	1.1E+09	67052.22	4.1E+10		
ship	732996	1.8E+12	8645006.83	8E+14		

The eight networks for which SP can perform belief update have an average size of the largest factor of less than 100,000 probabilities.

Table 3 shows the average computation cost (wall-clock time) of belief update by SP. In 14 out of 28 cases the time cost of SP using T_{fiw} is approximately the same as the time cost SP using $\hat{T}$. The difference is less than five per cent. Since SP using T_{MPD} is only able to solve the most simple networks, the time cost of SP in this case is low and in many cases comparable to the time cost of $\hat{T}$ and T_{fiw} (at least in absolute terms).

Table 3. Average time cost of belief update using SP (mean and standard deviation).

$\mathcal{N}$	$\mu(\hat{T})$	$\sigma(\hat{T})$	$\mu(T_{fiw})$	$\sigma(T_{fiw})$	$\mu(T_{MPD})$	$\sigma(T_{MPD})$
ADAPT_DX09_T2	0.19	0.011	0.19	0.011		
Amirali_network	0.38	0.043	0.61	0.835		
Heizung.	0.24	0.326	0.23	0.31		
Hepar_II	0.03	0	0.02	0	0.03	0
Mildew	0.04	0.001	0.05	0.006		
Munin1	0.74	3.26	0.98	7.408		
andes	0.13	0.005	0.11	0.004		
cc145	0.07	0.001	0.07	0.001	0.07	0.001
sacso	0.83	1.051	0.74	0.899		
ship	0.16	0.084	1.26	25.073		

6 Conclusion

This paper has investigated the impact of the secondary computational structure used for belief update on time and space performance of SP. The results of a preliminary empirical performance evaluation on a set of real-world Bayesian networks indicate that the tree structure used to control the message passing can have a significant impact on both time and space performance.

References

1. Butz, C.J., Oliveira, J.S., dos Santos, A.E., Madsen, A.L.: Bayesian network inference with simple propagation. In: Proceedings of the Twenty-Ninth International FLAIRS Conference (2016)
2. Jensen, F.V., Jensen, F.: Optimal junction trees. In: Proceedings of UAI, pp. 360–366 (1994)
3. Madsen, A.L.: Variations over the message computation algorithm of lazy propagation. IEEE Trans. Syst. Man Cybern. Part B **36**(3), 636–648 (2006)
4. Madsen, A.L., Butz, C.: On the tree structure used by lazy propagation for inference in Bayesian networks. In: van der Gaag, L.C. (ed.) ECSQARU 2013. LNCS, vol. 7958, pp. 400–411. Springer, Heidelberg (2013)
5. Madsen, A.L., Jensen, F.V.: Lazy propagation: a junction tree inference algorithm based on lazy evaluation. Artif. Intell. **113**(1–2), 203–245 (1999)
6. Shachter, R.D.: Evaluating influence diagrams. Oper. Res. **34**(6), 871–882 (1986)