ELSEVIER

Artificial Intelligence 102 (1998) 323–333

Artificial Intelligence

Some improvements to the Shenoy–Shafer and Hugin architectures for computing marginals

Tuija Schmidt, Prakash P. Shenoy *

School of Business, University of Kansas, Summerfield Hall, Lawrence, KS 66045-2003, USA

Received 5 January 1998; received in revised form 23 March 1998

Abstract

The main aim of this paper is to describe two modifications to the Shenoy–Shafer architecture with the goal of making it computationally more efficient in computing marginals of the joint valuation. We also describe a modification to the Hugin architecture. Finally, we briefly compare the traditional and modified architectures by solving a couple of small Bayesian networks, and conclude with a statement of further research. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Shenoy–Shafer architecture; Hugin architecture; Computing marginals

1. Introduction

In this paper we introduce some modifications to the Shenoy–Shafer [10] and Hugin architectures [2,3]. Although both architectures are valid more generally, we will describe our modifications for the case of Bayesian networks.

Our main modification to the Shenoy–Shafer architecture is the introduction of a new phase called "transfer of valuations." The transfer of valuations phase reduces the amount of calculations needed for computation of marginal probabilities. Our other modification to the Shenoy–Shafer architecture is the introduction of a new rule for computing marginals. Furthermore, we sketch a modification to the Hugin architecture where all combinations are done on a binary basis. Combining valuations on a binary basis reduces the amount of multiplications needed. Finally, we compare the modified and the traditional architectures in terms of computational efficiency, and conclude with a statement of further research.

An outline of the remainder of this article is as follows. In Section 2, we briefly describe the computational aspects of the Shenoy–Shafer (SS) architecture. In Section 3,

* Corresponding author. Email: pshenoy@ukans.edu.

0004-3702/98/$19.00 © 1998 Elsevier Science B.V. All rights reserved.
PII: S0004-3702(98)00047-2

we describe two modifications of the SS architecture designed to make it computationally more efficient. In Section 4, we briefly describe the computational aspects of the Hugin architecture. In Section 5, we describe a modification of the Hugin architecture. This modification was suggested by one of the authors of the Hugin architecture [1]. In Section 6, we compare the traditional architectures with our suggested modifications for two small problems to demonstrate the increase in computational efficiencies. Finally, in Section 7, we summarize our findings and conclude with a statement of further research.

2. The Shenoy–Shafer architecture

In this section we briefly describe the Shenoy–Shafer architecture. This architecture was first described by Shenoy and Shafer [10]. Recently, Shenoy [9] has described a new data structure, called binary join tree, designed to make this architecture more efficient. Lepar and Shenoy [7] do a detailed study of this architecture and compare it with the Lauritzen and Spiegelhalter [6] and Hugin architectures. We will describe the Shenoy–Shafer architecture using a small example called *Diabetes*.

Diabetes. Diabetes (D) causes two symptoms, blue toe (B) and Glucose in urine (G). A Bayesian network representation of this problem is shown in Fig. 1. The Bayesian network representation includes valuations δ for $\{D\}$ representing the prior probability distribution for D, β for $\{D, B\}$ representing the conditional probability distribution for B given D, and γ for $\{D, G\}$ representing the conditional probability distribution of G given D. Suppose we have evidence for B and G represented as valuations λ_B for $\{B\}$ and λ_G for $\{G\}$, respectively. The problem is to compute the marginals of the posterior joint distribution for D, B and G.

In the Shenoy–Shafer architecture, we start with a set of valuations denoted by ϑ. The combination of all valuations in ϑ is called the joint valuation, $\phi = \otimes\vartheta$. In the Diabetes problem, $\vartheta = \{\delta, \beta, \gamma, \lambda_B, \lambda_G\}$, and the joint valuation is the unnormalized posterior joint probability distribution of $\{D, B, G\}$. The problem is to find the marginal of the joint valuation for subsets of interest, i.e., $\phi^{\downarrow r}$, where r is a subset of the set of variables whose marginal is desired.

The domains of the valuations form a hypergraph H. In the Diabetes example, $H = \{\{D\}, \{D, B\}, \{D, G\}, \{B\}, \{G\}\}$.

The first phase in Shenoy–Shafer architecture is called the *join tree construction phase*. In this phase, we first arrange the subsets in H in a binary join tree [9] and then add

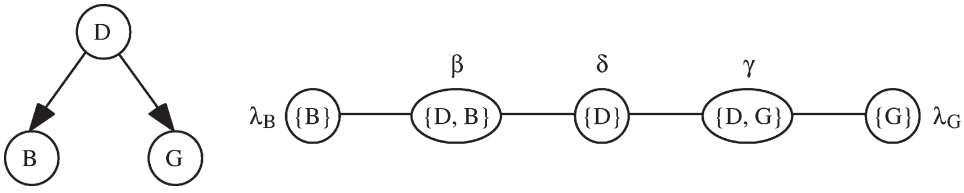

Fig. 1. A Bayesian network and a corresponding join tree for the Diabetes problem.

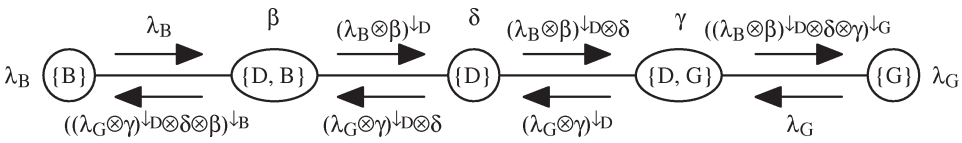

Fig. 2. The messages computed in the Shenoy–Shafer architecture.

singleton subsets for which we need marginals to the binary join tree if these are not already included [7]. After the binary join tree has been constructed, we associate each valuation in ϑ with the corresponding node in the binary join tree. For the Diabetes problem, a binary join tree with the associated valuations is shown in Fig. 1.

The second phase in the Shenoy–Shafer architecture is called the *message-passing phase*. In this phase, each node, upon receiving a request, sends a message to its neighbor that requests such a message. The nodes for which the marginals are desired request messages from their neighbors. The messages are computed by using the rule described in [7,10]. In the Diabetes problem, the messages sent between adjacent nodes are shown in Fig. 2. In this example, all eight messages are computed. If only the marginal for D was needed, then only half the messages (four of eight) would be computed.

The third and final phase is called the *marginal computation phase*. In this phase, we compute the marginals for the desired subsets using the following rule.

Rule for Computing Marginals. When a node r has received a message from each of its neighbors, it combines all messages together with its own valuation and reports the results as its marginal. If ϕ denotes the joint valuation, then

$$\phi^{\downarrow r} = \otimes\left\{\mu^{t\to r} \mid t \in N(r)\right\} \otimes \alpha_r.$$

In the Diabetes problem, node B computes

$$\lambda_B \otimes \mu^{DB\to B} = \lambda_B \otimes \left((\lambda_G \otimes \gamma)^{\downarrow B} \otimes \delta \otimes \beta\right)^{\downarrow B},$$

node D computes

$$\delta \otimes \mu^{DB\to D} \otimes \mu^{DG\to D} = \delta \otimes (\lambda_B \otimes \beta)^{\downarrow D} \otimes (\lambda_G \otimes \gamma)^{\downarrow D},$$

and node G computes

$$\lambda_G \otimes \mu^{DG\to G} = \lambda_G \otimes \left((\lambda_B \otimes \beta)^{\downarrow D} \otimes \delta \otimes \gamma\right)^{\downarrow G}.$$

These are the respective (unnormalized) marginals for the singleton subsets $\{D\}$, $\{B\}$ and $\{G\}$. This completes our brief description of the computational aspects of the Shenoy–Shafer architecture.

3. Two modifications to the Shenoy–Shafer architecture

In this section we describe the two modifications to the Shenoy–Shafer architecture. These modifications are motivated by the presence of repetitive combinations in the

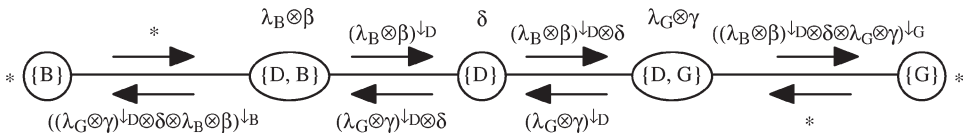

Fig. 3. The messages in the Shenoy–Shafer architecture after transfer of valuations ($*$ denotes the identity valuation).

traditional Shenoy–Shafer architecture. We start by motivating the modifications and then describe the actual modifications.

Consider again the Diabetes problem. Notice that although we calculate $\lambda_G \otimes \gamma$ in the message from $\{D, G\}$ to $\{D\}$, we combine λ_G and γ again during the computation of the marginal for G: $\phi^{\downarrow G} = \lambda_G \otimes (\mu^{D \to DG} \otimes \gamma)^{\downarrow G}$. Similarly, the combination $\lambda_B \otimes \beta$ is done twice—once during the computation of the message from $\{D, B\}$ to $\{B\}$, and once during the computation of the marginal for B. These inefficiencies can be removed by recursively transferring the valuations from the leaves to the adjacent supersets before the message-passing phase. It may seem counterintuitive to transfer a valuation from a subset to its superset. However, since the combination of the transferred valuation happens on the state space of the superset anyway, there is no computational penalty in doing the transfer. In the Diabetes example, we transfer the valuation λ_B from $\{B\}$ to $\{D, B\}$, and transfer λ_G from G to $\{D, G\}$.

Notice that since $\{D\}$ is not a leaf node (and never becomes one), we do not transfer its associated valuation δ anywhere. In Fig. 2, notice that the combination of δ happens on the state space of $\{D\}$. If we transfer δ to either $\{D, B\}$ or $\{D, G\}$, there would be a computational penalty since the combination of δ would then happen on a larger state space.

If we do the transfer of valuations prior to the message-passing phase and then do the message-passing, the computations for the Diabetes problem are shown in Fig. 3. Notice, that the combinations $\lambda_B \otimes \beta$ and $\lambda_G \otimes \gamma$ are now done just once. Since $\{B\}$ and $\{G\}$ have no valuations associated with them, the messages to these nodes from their neighbors are the marginals for these nodes, and no combination is involved in the computation of the marginals for these nodes during the marginal computation phase (unlike the case in Fig. 2 where no valuations were transferred). In Section 6, we will return to this example and discuss the computational savings as a result of this modification.

The second modification is motivated by the following observation. Notice that, when D computes its marginal in Fig. 2, it computes

$$\delta \otimes \mu^{DB \to D} \otimes \mu^{DG \to D} = \delta \otimes (\lambda_B \otimes \beta)^{\downarrow D} \otimes (\lambda_G \otimes \gamma)^{\downarrow D}.$$

However, the combination $\delta \otimes \mu^{DB \to D}$ was already done before in the message from D to $\{D, G\}$. Thus we could have avoided this repetition by computing $\mu^{D \to DG} \otimes \mu^{DG \to D}$ instead of $\delta \otimes \mu^{DB \to D} \otimes \mu^{DG \to D}$. Alternatively, we could have computed the marginal for D by computing $\mu^{D \to DB} \otimes \mu^{DB \to D}$. We can use this rule as long as the neighbor of the node whose marginal is desired is a superset of the node because when a node sends a message to a neighbor that is its superset, there is no marginalization involved in the computation of the message. Thus, for example, we cannot use this rule for computing the marginal for $\{D, B\}$—we have to use the old rule.

We will now formally describe the two modifications of the Shenoy–Shafer architecture. In the modified Shenoy–Shafer architecture, the computation of marginals is done in four phases (instead of three in the traditional Shenoy–Shafer architecture). Following the join tree construction phase and before the message-passing phase, we introduce a new phase called the *transfer (of valuations) phase*.

In the transfer phase, valuations associated with some of the nodes of the join tree (constructed at the end of the join tree construction phase) are transferred to other nodes. The precise rules for transfer of valuations are as follows:

Rules for transferring valuations

Assume that initially each node has a list of its neighbors. Let $n(r)$ denote such a list for node r. We call $n(r)$ the *neighbor set* of node r. This list gets modified during the transfer of valuations phase. However, the structure of the tree is not modified.

Rule 1. If $|n(r)| = 1$, say $n(r) = \{s\}$, and $r \subseteq s$, then r sends the valuation associated with it to s, replaces the valuation associated with it by the identity valuation, and deletes s from its neighbor set (so $n(r)$ is now the empty set).

Rule 2. If a node r receives a valuation, say α_s, from its neighbor s, then r replaces the valuation associated with it, say α_r, by $\alpha_r \otimes \alpha_s$, and removes s from its neighbor set, $n(r) \leftarrow n(r) \backslash \{s\}$.

The valuations transferred between nodes in this phase are not stored in the separators. This phase ends when there are no nodes that satisfy the condition stated in Rule 1 ($n(r) = \{s\}$, and $r \subseteq s$).

In the Diabetes problem, node $\{B\}$ has only one neighbor that is its superset. Hence, it transfers λ_B to $\{D, B\}$ and replaces λ_B by the identity valuation. Node $\{D, B\}$ replaces its own valuation β by the combination $\lambda_B \otimes \beta$ and removes node B from its neighbor set. Node $\{G\}$ also has one neighbor that is its superset. Hence, it sends its valuation λ_G to it, and replaces λ_G by the identity valuation. Node $\{D, G\}$ replaces its valuation γ by the combination $\lambda_G \otimes \gamma$. Nodes $\{D, B\}$ and $\{D, G\}$ now have only one element in their neighbor set. However, because neither of them have neighbors that are supersets of them, the transfer of valuations phase ends. At the end of the transfer of valuation phase, the valuations associated with each node are as shown in Fig. 3. Notice, that the structure of the join tree has not changed. Only the valuations have been transferred.

The second modification of the Shenoy–Shafer method applies to calculation of the marginals. In the traditional Shenoy–Shafer method the rule for computing marginals is as follows: when a node r has received a message from each of its neighbors, it combines all messages together with its own valuation and reports the results as its marginal. This can lead to up to three combinations as a node can receive up to three messages in a binary join tree. A modified rule for computing marginals is as follows.

Modified Rule for Computing Marginals. Suppose s is a neighbor of r such that $s \supseteq r$. Node r computes its marginal by combining the message from r to s by the message from s to r (regardless of the number of other messages node r receives), i.e.,

$$\phi^{\downarrow r} = \mu^{s \to r} \otimes \mu^{r \to s}.$$

The reason this rule works is as follows. Let $N(r)$ denote the set of neighbors of node r. (Notice that $N(\cdot)$ is strictly a function of the topology of the join tree and is different from $n(\cdot)$ that got modified during the transfer phase.)

$$\begin{aligned}\phi^{\downarrow r} &= \otimes\{\mu^{t\to r} \mid t \in N(r)\} \otimes \alpha_r \\ &= \mu^{s\to r} \otimes \{\otimes\{\mu^{t\to r} \mid t \in N(r)\backslash\{s\}\} \otimes \alpha_r\} \\ &= \mu^{s\to r} \otimes \mu^{r\to s}.\end{aligned}$$

Notice that the modified rule only applies to nodes that have a neighbor that is its superset. If this condition is not satisfied, i.e., none of the neighbors of a node are its supersets, then such a node has to use the old rule for computing its marginal. Typically, we are interested in marginals for singletons. In such cases, we can always apply this rule since a singleton will always have a superset as its neighbor. Notice, that if a singleton node is a leaf in the join tree, then it will have the identity valuation associated with it after the transfer of valuation phase. In this case, the marginal for this singleton variable will simply be the message that it receives from its neighbor and there is no combination involved. This is the case for nodes $\{B\}$ and $\{G\}$ in Fig. 3.

4. The Hugin architecture

In this section, we briefly describe the computational aspects of the Hugin architecture. The Hugin architecture has its roots in the method proposed by Lauritzen and Spiegelhalter [6] for computing marginals of probability distributions. It was proposed by Jensen et al. [2,3] and is incorporated in the software product Hugin. Recently, Lauritzen and Jensen [5] have described some axioms underlying this architecture so it applies not only for probabilities but also to any domain that satisfies the axioms.

The Hugin architecture also has three phases. The first phase is the join tree construction phase. The join tree constructed is called a *junction tree*. Fig. 4 shows a junction tree for the Diabetes problem. This junction tree has two cliques, $\{D, B\}$ and $\{D, G\}$, and one separator, $\{D\}$. Valuations λ_B, β and δ are associated with $\{D, B\}$, and valuations λ_G and γ are associated with $\{D, G\}$.

The second phase is the message-passing phase. In this phase, any one clique in the join tree is designated as the root. The message-passing is done in two stages: the inward pass (in which each non-root node sends a message to its inward neighbor, i.e., the neighbor toward the root) and the outward pass (in which each non-leaf node sends a message to its outward neighbor, i.e., the neighbor away from the root). The rules for passing messages (in both stages) are described in [2,3,8].

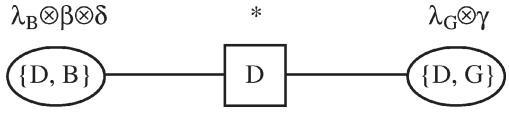

Fig. 4. The junction tree with associated valuations for the Diabetes problem.

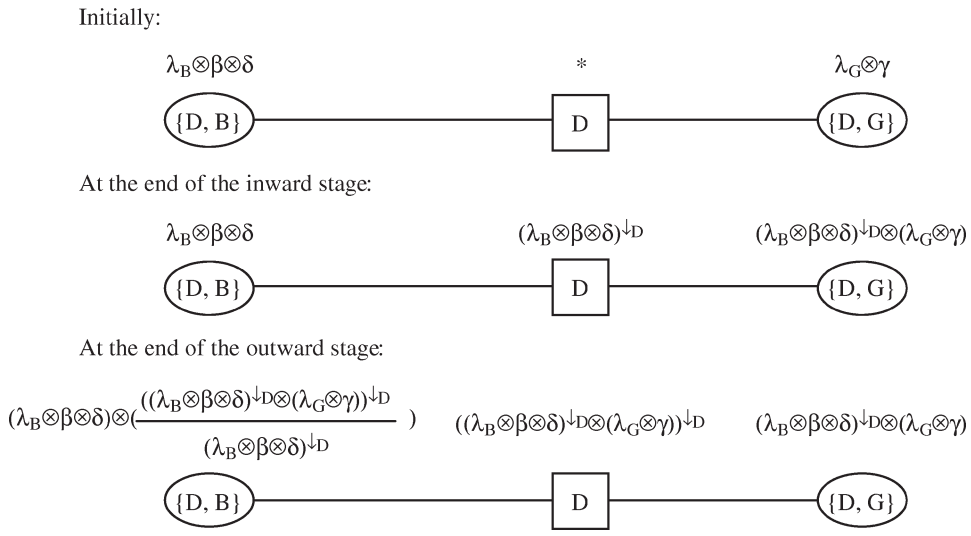

Fig. 5. The details of the Hugin message-passing phase for Diabetes. Node $\{D, G\}$ is assumed to be the root.

The message-passing phase ends when each clique has sent a message to and received a message from each of its neighbors. At the end of the message-passing phase, the valuation associated with each clique and with each separator is the marginal of the joint for the clique or separator. Fig. 5 shows the details of the message-passing phase for the Diabetes problem where clique $\{D, G\}$ is designated as the root.

The third phase is the computation of marginals phase. The needed marginals are computed from the marginals of a smallest separator or a smallest clique that contains the subset. For example, in the Diabetes problem, the marginal for $\{D\}$ is computed from the marginal for the separator (since the separator is $\{D\}$, no computation is involved,

$$\phi^{\downarrow D} = \big((\lambda_B \otimes \beta \otimes \delta)^{\downarrow D} \otimes (\lambda_G \otimes \delta)\big)^{\downarrow D},$$

the marginal for $\{B\}$ is computed from the marginal for $\{D, B\}$,

$$\phi^{\downarrow B} = \big(\phi^{\downarrow\{D,B\}}\big)^{\downarrow B},$$

and the marginal for $\{G\}$ is computed from the marginal for $\{D, G\}$, $(\phi^{\downarrow\{D,G\}})^{\downarrow G}$.

5. A modification to the Hugin architecture

In this section we sketch a modification to the traditional Hugin architecture. This modification is based on a discussion with Finn V. Jensen [1], one of the authors of the Hugin architecture. It is motivated by the same considerations that led to the notion of binary join trees in the Shenoy–Shafer architecture [9]. To motivate the modification, we will describe another problem, the Chest Clinic problem from Lauritzen and Spiegelhalter [6].

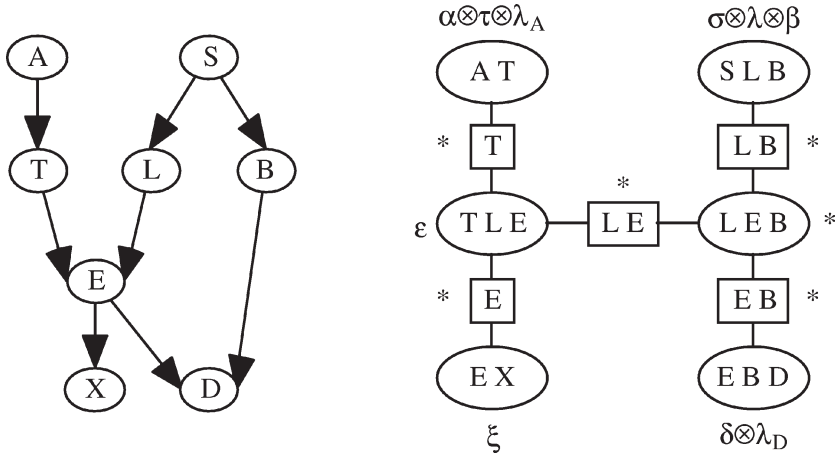

Fig. 6. The Bayesian network and a junction tree for the Chest Clinic problem.

Chest Clinic Problem. Shortness-of-breath (dyspnoea) may be due to tuberculosis (T), lung cancer (L) or bronchitis (B), or none of them, or more than one of them. A recent visit to Asia (A) increases the chances of tuberculosis, while smoking (S) is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray (X) do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea (D). E denotes "has either tuberculosis or lung cancer". Fig. 6 shows the Bayesian network and a junction tree with associated valuations for the Chest Clinic problem.

In the Chest Clinic problem, the valuation associated with clique $\{S, L, B\}$ is $\sigma \otimes \lambda \otimes \beta$. If each variable has a state space of 10 values, then the computation of $\sigma \otimes \lambda \otimes \beta$ on the state space of clique $\{S, L, B\}$ requires 2000 multiplications. If the valuations are combined on a binary basis, the amount of multiplications is reduced. Multiplying valuations on a binary basis is done by first combining two valuations with the smallest domains. Therefore, we first combine $\sigma \otimes \lambda$ on the state space of $\{S, L\}$ requiring 100 multiplications, and then combine $\sigma \otimes \lambda$ with β on the state space of $\{S, L, B\}$ requiring an additional 1000 multiplications for a total of 1100 multiplications, a savings of 900 multiplications. Furthermore, multiplications can also be saved during the inward stage of the message-passing phase if messages are combined on a binary basis. For example, in the Chest Clinic problem, node $\{T, L, E\}$ combines its own valuation ε with the two messages it receives from its neighbors. Assuming that each variable has a state space of 10 values, the message from node $\{L, E, B\}$ has a state space of 100 values, the message from node $\{E, X\}$ has a state space of 10 values, and ε has a state space of 1000 values. Therefore, combining all three messages on the state space of clique $\{T, L, E\}$ results in 2000 multiplications. If the messages are combined on a binary basis, the two messages from nodes $\{E, X\}$ and $\{L, E, B\}$ are combined first, which requires 100 multiplications. This product is then combined with ε requiring 1000 multiplications, a total of 1100 multiplications. This results in a savings of 900 multiplications.

6. A comparison

In this section, we will compare the modified Shenoy–Shafer and Hugin architectures with the traditional Shenoy–Shafer and Hugin architectures, respectively. We will base our comparison on two small problems. The first problem is the Diabetes problem described in Section 2. The second problem is Lauritzen and Spiegelhalter's [6] Chest Clinic problem described in Section 5. The computations in the Shenoy–Shafer architecture for the Chest Clinic problem are based on the binary join tree that results from the construction procedure described in [7].

In both problems, we will assume that each variable has a state space of 10 values. In probability theory, combination of two valuations on the state space of n variables will involve 10^n multiplications. Marginalizing a valuation for a domain consisting of n variables to a domain consisting of $n-1$ variables will involve $9 \times 10^{n-1}$ additions. And computing a ratio of two valuations on a domain consisting of n variables will involve 10^n divisions. Table 1 shows the number of additions, multiplications, and divisions needed for the Chest Clinic example with both the traditional and modified architectures.

First notice that the number of additions done in the modified Shenoy–Shafer architecture is the same as that in the traditional Shenoy–Shafer architecture. The same is true for the Hugin architecture. This is always true because the number of additions is strictly a function of the topology of the join tree. Since the transfer of valuations phase does not change the topology of the join tree, it has no impact on the number of additions done. Therefore, the modifications suggested here reduce only the number of multiplications.

The modified Shenoy–Shafer architecture uses 430 multiplications to solve the Diabetes example, and 9730 multiplications to solve the Chest Clinic example with evidence. This is 30 less multiplications in the Diabetes example and 1330 less multiplications in the Chest

Table 1
Comparison of the modified architectures with the corresponding traditional ones

# of binary arithmetic operations	Shenoy–Shafer		Hugin	
Problem	Traditional	Modified	Traditional	Modified
Diabetes with evidence for B and G				
# binary additions	360	360	360	360
# binary multiplications	460	430	500	500
# binary divisions			10	10
Total	820	790	870	870
Chest Clinic with evidence for A and D				
# binary additions	9900	9900	9900	9900
# binary multiplications	11060	9730	10400	8510
# binary divisions			320	320
Total	20960	19630	20620	18730

Clinic example than in the traditional Shenoy–Shafer architecture. Most of the savings occur during the transfer of valuations phase. Because the valuations for leaf nodes are stored in their superset neighbors, repetitive multiplications are eliminated.

In general, the two modifications described here will always reduce the number of multiplications done in the traditional Shenoy–Shafer architecture. The essence of the two proposed modifications is to eliminate computations that are repeated. The amount of savings will depend on the specifics of the Bayesian network under consideration. The transfer phase modifications will result in more savings than the modified rule for computing marginals. Both modifications will add some minimal overhead to the computational process, but the savings will far outweigh the overhead.

The modified Hugin method uses a total of 18730 arithmetic operations when solving Chest Clinic example with evidence for A and D. This is 1890 less multiplications than in the traditional Hugin method. 990 of the multiplication savings are achieved in the beginning, when valuations for cliques are combined on a binary basis. Instead of computing $\alpha \otimes \lambda_A \otimes \tau$ on the state space of node $\{A, T\}$ and computing $\sigma \otimes \lambda \otimes \beta$ on the state space of node $\{S, L, B\}$, we combine the valuations with the smallest domains first, e.g., $\alpha \otimes \lambda_A$ on the state space of node $\{A\}$, and then $(\alpha \otimes \lambda_A) \otimes \tau$ on the state space of node $\{A, T\}$. We do the same for $(\sigma \otimes \lambda) \otimes \beta$. The other 900 savings occur during the inward pass at clique $\{T, L, E\}$, when its valuation ε and the messages from nodes $\{E, X\}$ and $\{L, E, B\}$ are combined on a binary basis as described in Section 5.

In general, the modification described here will always reduce the number of multiplications done in the traditional Hugin architecture. The exact savings will depend on the specifics of the Bayesian network under consideration, but the potential for savings is big especially for problems in which the junction trees are non-binary [7]. Implementing binary multiplications will result in more storage space, so we are achieving savings in time at the expense of space.

7. Conclusions

The main objective of this paper is to describe some modifications to the Shenoy–Shafer and Hugin architectures. We sketch both methods and describe the modifications to them. We also briefly compare the computational efficiency of the modified and traditional methods.

The introduction of the transfer phase and the modified rule for computing marginals always reduces the amount of computations needed in Shenoy–Shafer method. As a result of these changes, the modified Shenoy–Shafer architecture is always more computationally efficient than the traditional Shenoy–Shafer method. The Hugin method is also improved by introducing the binary Hugin method that is always more efficient than the traditional Hugin method.

A question not addressed in this paper is the relative efficiencies of the modified Shenoy–Shafer and modified Hugin architectures. A detailed comparison of the traditional Shenoy–Shafer and traditional Hugin architectures is found in [7]. Comparison of the modified architectures is a task that remains to be done.

Recently, Kjærulff [4] has suggested an alternative modification to the Hugin architecture called "nested junction trees." Nested junction trees are data structures where some cliques in a junction tree are substituted by junction trees that represent the details of the computation done in the cliques. This strategy can also be used for the Shenoy–Shafer architecture. A detailed comparison of this strategy with the binary join tree strategy also remains to be done in both architectures.

Acknowledgements

The authors are grateful to Vasilica Lepar for comments and discussion of the paper.

References

[1] F.V. Jensen, private communication, 1996.
[2] F.V. Jensen, K.G. Olesen, S.K. Andersen, An algebra of Bayesian belief universes for knowledge-based systems, Networks 20 (5) (1990) 637–659.
[3] F.V. Jensen, S.L. Lauritzen, K.G. Olesen, Bayesian updating in causal probabilistic networks by local computation, Comput. Statist. Quarterly 4 (1990) 269–282.
[4] U. Kjærulff, Nested junction trees, in: D. Geiger, P.P. Shenoy (Eds.), Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference, Morgan Kaufmann, San Francisco, CA, 1997, pp. 294–301.
[5] S.L. Lauritzen, F.V. Jensen, Local computation with valuations from a commutative semigroup, Technical Report No. R-96–2028, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1996.
[6] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems (with discussion), J. Royal Statist. Soc. Ser. B 50 (2) (1988) 157–224.
[7] V. Lepar, P.P. Shenoy, A comparison of architectures for exact computation of marginals, Working Paper No. 274, School of Business, University of Kansas, Lawrence, KS, 1997.
[8] G. Shafer, Probabilistic Expert Systems, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.
[9] P.P. Shenoy, Binary join trees for computing marginals in the Shenoy–Shafer architecture, Internat. J. Approx. Reasoning 17 (2–3) (1997) 239–263.
[10] P.P. Shenoy, G. Shafer, Axioms for probability and belief-function propagation, in: R.D. Shachter, T.S. Levitt, J.F. Lemmer, L.N. Kanal (Eds.), Uncertainty in Artificial Intelligence, Vol. 4, North-Holland, Amsterdam, 1990, pp. 169–198. Reprinted in: G. Shafer, J. Pearl (Eds.), Readings in Uncertain Reasoning, Morgan Kaufmann, San Mateo, CA, 1990, pp. 575–610.