SCHOOL OF BUSINESS WORKING PAPER NO. 274

A COMPARISON OF ARCHITECTURES FOR EXACT COMPUTATION OF MARGINALS

Vasilica Lepar and Prakash P. Shenoy

February 1997†

Institute of Informatics
University of Fribourg
Site Regina Mundi
Rue Faucigny 2
CH-1700, Fribourg, Switzerland
vasilica.lepar@unifr.ch

School of Business
University of Kansas
Summerfield Hall
Lawrence, KS 66045-2003, USA
pshenoy@ukans.edu

† Comments and suggestions for improvement are welcome and will be gratefully appreciated.

TABLE OF CONTENTS

ABSTRACT.....1

1 INTRODUCTION.....1

2 BAYESIAN NETWORK MODELS & THREE PROBLEMS.....2

2.1 Bayesian Network Models.....2

2.2 The Chest Clinic Problem.....3

2.2 The Stud Farm Problem.....4

2.3 The Genetic Reproduction Problem.....5

3 THE LAURITZEN-SPIEGELHALTER ARCHITECTURE.....7

4 THE HUGIN ARCHITECTURE.....12

5 THE SHENOY-SHAFER ARCHITECTURE.....16

6 COMPARISON.....19

ACKNOWLEDGMENTS.....28

REFERENCES 28

APPENDIX. COUNTING STORAGE AND OPERATIONS.....30

SELECTED WORKING PAPERS.....57

A COMPARISON OF ARCHITECTURES FOR EXACT COMPUTATION OF MARGINALS

Vasilica Lepar and Prakash P. Shenoy

ABSTRACT

In the last decade, several architectures have been proposed for exact computation of marginals using local computation. In this paper we compare three architectures—Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer—from the perspective of graphical structure for message propagation, message-passing scheme, storage efficiency, and computational efficiency.

Key Words: Lauritzen-Spiegelhalter architecture, Hugin architecture, Shenoy-Shafer architecture, computing marginals

---

1 INTRODUCTION

In the last decade, several architectures have been proposed for exact computation of marginals of multivariate discrete probability distributions. One of the pioneering architectures for computing marginals was proposed by Pearl [1986]. Pearl's architecture applies to singly connected Bayes nets. For multiply connected Bayes nets, Pearl [1986] proposed the method of conditioning to reduce a multiply connected Bayes net to several singly connected Bayes nets.

In 1988, Lauritzen and Spiegelhalter [1988] proposed an alternative architecture for computing marginals that applies to any Bayes net. Subsequently, Jensen *et al*. [1990a, b] proposed a modification of the Lauritzen-Spiegelhalter architecture. We call this architecture the Hugin architecture since this architecture is implemented in Hugin, a software tool developed by the same group. Recently, this architecture has been abstracted by Lauritzen and Jensen [1996] so that it applies more generally to other domains including the Dempster-Shafer's belief function theory.

Inspired by the work of Pearl, Shenoy and Shafer [1986] first adapted and generalized Pearl's architecture to the case of finding marginals of joint Dempster-Shafer belief functions in join trees. Later, inspired by the work of Lauritzen and Spiegelhalter [1988] for the case of probabilistic reasoning, they proposed an abstract framework for computing marginals in join trees that applies to any domain satisfying some axioms [Shenoy and Shafer 1990]. We refer to this architecture as the Shenoy-Shafer architecture. In a sense, the Shenoy-Shafer architecture can be considered as an adaptation of Pearl's propagation scheme to the join tree graphical structure. Recently, Shenoy [1997] has proposed a refinement of join trees, called binary join trees, designed to improve the computational efficiency of the Shenoy-Shafer architecture.

In this paper, we compare the Lauritzen-Spiegelhalter (LS), Hugin, and Shenoy-Shafer (SS) architectures from the perspective of graphical structure for message propagation, the message passing scheme, storage efficiency, and computational efficiency.

Our main findings are as follows. The Hugin architecture is more computationally efficient than the LS architecture, and less storage efficient than the LS architecture. This is not surprising. What is surprising is that we are unable to make any general statements regarding the relative storage efficiencies of the LS and SS architectures, or the relative computational efficiencies of the Hugin and SS architectures. For some problems, LS has less storage than SS, and for some problems, SS has less storage than LS. For some problems, Hugin is more computationally efficient than SS and for some problems, SS is more computationally efficient than Hugin. We identify some aspects of the Hugin architecture that are better than SS, and some aspects of the SS architecture that are better than Hugin. Hopefully, this will lead to improvements in both architectures.

2 BAYESIAN NETWORK MODELS & THREE PROBLEMS

In this section, we will define a Bayesian network probability model and then describe three problems: Lauritzen and Spiegelhalter's [1988] *Chest Clinic* (CC) problem, Jensen's [1996] *Stud Farm* (SF) and *Genetic Reproduction* (GR) problems. We will use the Chest Clinic problem to illustrate the three architectures. We will compare the efficiencies of the three architectures using all three problems. We start by defining a Bayesian network model.

2.1 Bayesian Network Models

First we introduce our notation. We denote variables by uppercase Roman alphabets, A, B, C, etc., and the set of all variables by Ψ. We denote subsets of variables by lowercase Roman alphabets c, s, t, etc. We denote the set of possible states of a variable X by Ω_X, and we assume that the set of possible states of a subset c of variables is the Cartesian product of the state space of individual variables in the subset c, $\Omega_c = \times\{\Omega_X \mid X \quad c\}$. We denote states of a subset of variables by lowercase boldfaced letters such as $\mathbf{x}$, $\mathbf{y}$, etc. If $\mathbf{x}$ is a state of c and b c, then $\mathbf{x}^{\downarrow b}$ denotes the projection of $\mathbf{x}$ to b obtained by simply dropping states of variables in c \ b. Of course, $\mathbf{x}^{\downarrow b}$ Ω_b.

Suppose c is a subset of variables. A *potential* for c is a function $\chi: \Omega_c \rightarrow \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of non-negative real numbers. We call c the *domain* of potential χ. We will denote potentials by lowercase Greek letters.

We define multiplication of potentials as follows. Suppose α is a potential for a, and suppose β is a potential for b. Then $\alpha \quad \beta$, read as α times β, is a potential for a b defined as follows: $(\alpha \quad \beta)(\mathbf{x}) = \alpha(\mathbf{x}^{\downarrow a})\, \beta(\mathbf{x}^{\downarrow b})$ for all $\mathbf{x}$ $\Omega_{a \ b}$.

We define marginalization of potentials as follows. Suppose α is a potential for a and suppose b a. Then the *marginal* of α to b, denoted by $\alpha^{\downarrow b}$, is a potential for b defined as follows: $\alpha^{\downarrow b}(\mathbf{x}) = \Sigma\{\alpha(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} \quad \Omega_{a \setminus b}\}$ for all $\mathbf{x} \quad \Omega_b$.

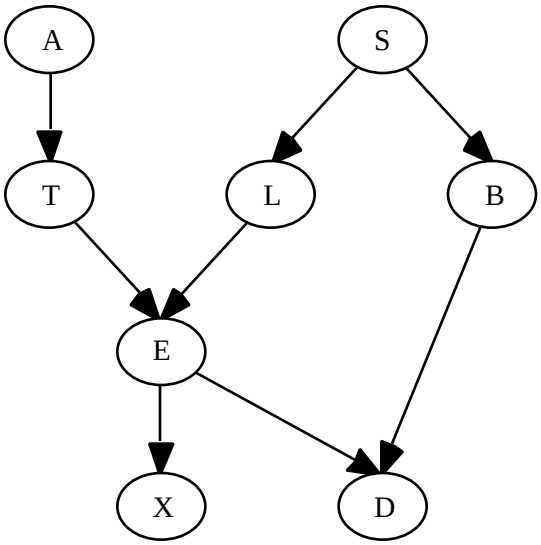
A *Bayesian network* model consists of a connected acyclic digraph G = (Ψ, Δ), and a set of conditional potentials $\{\kappa_V\}_{V \quad \psi}$, where Ψ represents the set of variables and Δ denotes the set of directed arcs between pairs of variables. An acyclic digraph is a finite oriented graph with no multiple arcs, and no directed cycles. If V and W are variables in Ψ and there is a directed arc from W to V, written as $W \rightarrow V$, then we say V is a *child* of W, and W is a *parent* of V. Let Pa(V) = $\{W \quad \Psi \ çW \rightarrow V\}$ denotes the set of *parents* of V. The conditional potentials $\{\kappa_V\}_{V \quad \psi}$ satisfy the following condition: κ_V: $\Omega_{\{V\} \quad Pa(V)} \rightarrow \mathbb{R}^+$ is such that $\kappa_V^{\downarrow Pa(V)}(\mathbf{x}) = 1$ for every $\mathbf{x} \quad \Omega_{Pa(V)}$. The assumption underlying a Bayesian network model is that the prior joint probability distribution P(Ψ) is given by P(Ψ) = $\{\kappa_V \mid V \quad \Psi\}$ For a more detailed description of a Bayesian network model, see [Lauritzen and Spiegelhalter 1988, and Pearl 1986].

2.2 The Chest Clinic Problem

In this section, we will first describe Lauritzen and Spiegelhalter's [1988] hypothetical Chest Clinic problem, and next, a Bayesian network model for it.

> Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea.

Figure 1. The Bayesian Network for the Chest Clinic Problem



This problem is modeled as a Bayesian network as shown in Figure 1. In this network, A denotes the variable visit to Asia?, S denotes Smoker?, T denotes Has Tuberculosis?, L denotes Has Lung Cancer?, B denotes Has Bronchitis?, E denotes Has Either Tuberculosis or Lung Cancer, X denotes Has positive X-ray?, and D denotes Has dyspnoea?. We assume that all variables are binary. Assessments are given in Table 1, representing a fictitious population coming to a chest clinic. Our notation uses *a* to indicate a positive response on the node A `visit to Asia?', ¬ *a* to indicate a negative response, and p(*a*) to stand for Pr(A = a). Similarly, *t* stands for the presence of `tuberculosis'; *s*, `smoker'; *l*, `lung cancer'; *b*, `bronchitis'; *e* `lung cancer or bronchitis'; *x*, `positive X-ray'; and *d*, `dyspnoea'. The probability tables for negative responses may be derived from Table 1.

Table 1. Conditional Probability Tables for the Chest Clinic Problem

α:	p(*a*) = .01	ε:	p(*e* | *l*, *t*) = 1 p(*e* | *l*, ¬*t*) = 1 p(*e* | ¬*l*, *t*) = 1 p(*e* | ¬*l*, ¬*t*)= 0
τ:	p(*t* | *a*) = .05 p(*t* | ¬*a*) = .01		
σ:	p(s) = .50	ζ:	p(*x* | *e*) = .98 p(*x* | ¬*e*) = .05
λ:	p(*l* | *s*) = .10 p(*l* | ¬*s*) = .01		
β:	p(*b* | *s*) = .60 p(*b* | ¬*s*) = .30	δ:	p(*d* | *e*, *b*) = .90 p(*d* | *e*, ¬*b*) = .70 p(*d* |¬*e*, *b*) = .80 p(*d* |¬*e*, ¬*b*)= .10

2.2 The Stud Farm Problem

The Stud Farm problem is taken from Jensen [1996].

The stallion Brian has sired Dorothy with the mare Ann and sired Eric with the mare Cecily. Dorothy and Fred are the parents of Henry, and Eric has sired Irene with Gwenn. Ann is the mother of both Fred and Gwenn, but their fathers are in no way related. The colt John with the parents Henry and Irene has been born recently; unfortunately, it turns out that John suffers from a life threatening hereditary disease carried by a recessive gene. The disease is so serious that John is displaced instantly, and as the stud farm wants the gene out of the production, Henry and Irene are taken out of breeding. What are the probabilities for the remaining horses to be carriers of the recessive gene?

Figure 2. The Bayesian Network for the Stud Farm Problem

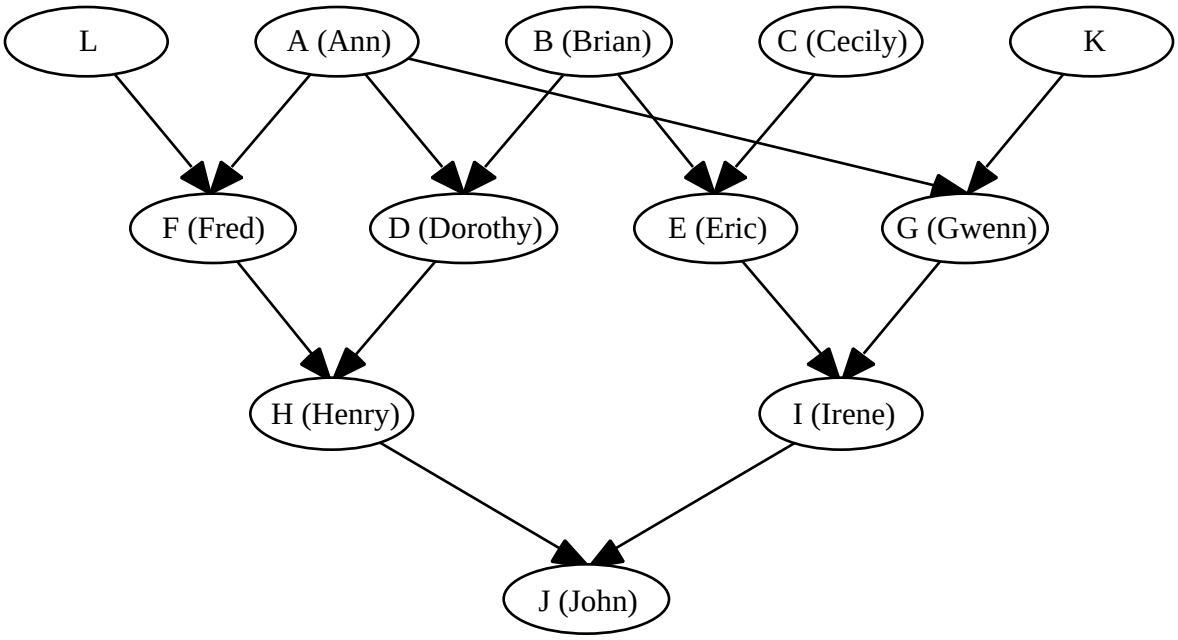


Table 2. Conditional Probability Tables for the *Stud Farm* Problem

α_L:	$p(l) = 0.99$	α_F:	$p(f \mid l, a) = 1$
α_J	$p(j \mid h, i) = 1$		$p(f \mid l, \neg a) = .50$
	$p(j \mid h, \neg i) = 0.5$		$p(f \mid \neg l, a) = .50$
	$p(jc \mid h, \neg i) = 0.5$		$p(f \mid \neg l, \neg a) = .25$
	$p(j \mid \neg h, i) = 0.5$		
	$p(jc \mid \neg h, i) = 0.5$		
	$p(j \mid \neg h, \neg i) = 0.25$		
	$p(jc \mid \neg h, \neg i) = 0.50$		

Assessments are given in Table 2. Our notation uses *l* to indicate a positive response on the node L 'is L pure?', ¬*l* to indicate a negative response, i.e., L is a carrier, and p(*l*) stand for Pr(L = *l*). For each node with no parents—A, B, C, K—we have the same probability as for L, and the priors are denoted by α_A, α_B, α_C, and α_K. The probability for a positive response on the node 'is F pure if its parents L and A are pure' is $p(f \mid l, a)$. For each node in this Bayesian network (except J) with two parents—D, E, G, H, I—we have the same conditional probability as for the node F respectively α_D, α_E, α_G, α_H, α_I. The probability tables for negative responses may be derived from Table 2. Node J has 3 states *j*, *jc*, *js*—*j* denotes John is pure, *jc* denotes John is a carrier, and *js*

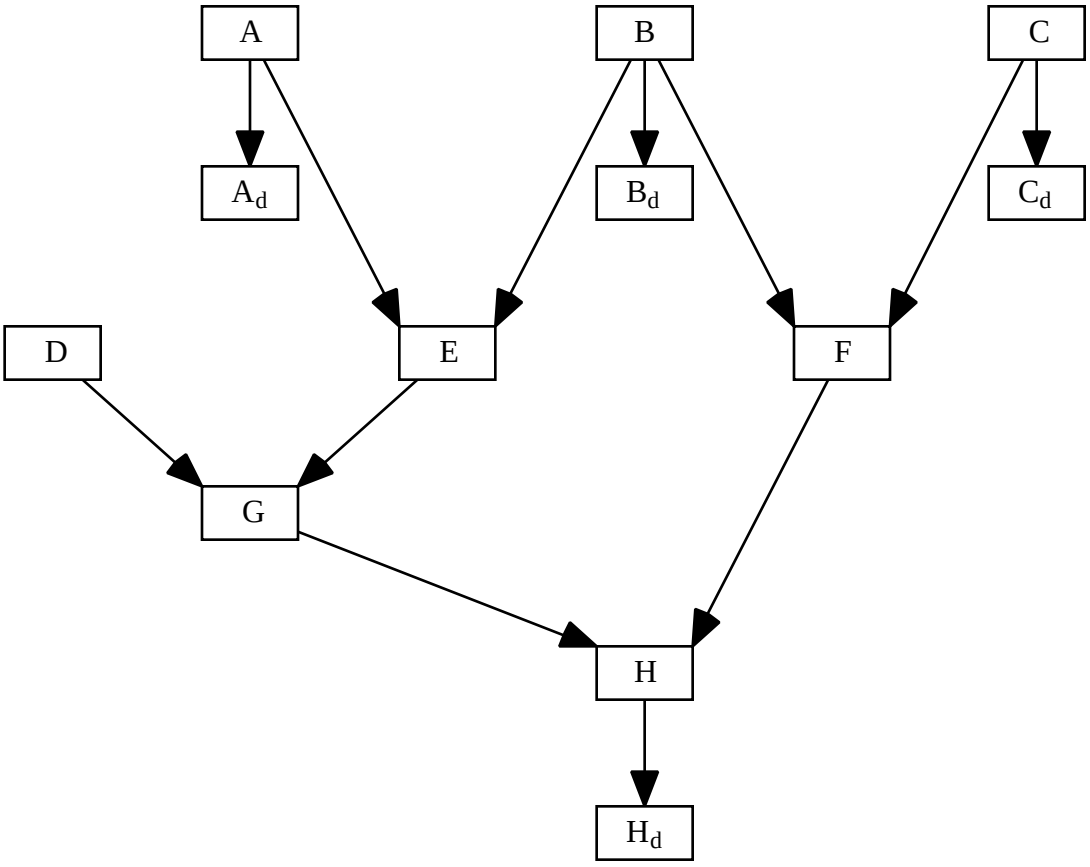
denotes John is sick. The conditional for J, α_J, is given in Table 2. Finally note that we have the observation o_J that John is sick, i.e., J = *js*.

2.3 The Genetic Reproduction Problem

The *Genetic Reproduction* (GR) problem is from the field of genetics. The problem concerns breeding and would typically be found in animal husbandry, but in order to make it more interesting, we will state it in terms of human beings:

> Florence (F) and Gregory (G) are about to reproduce. However, Gregory is Florence's nephew, and in the annals of Bartholomew (B), the father of Florence and grandfather of Gregory, a life-threatening disease has haunted. The disease is caused by a dominant allele a_1, and appears in a rather late stage of the individual's life-time. Bartholomew married twice, hence Florence and Gregory's mother are half-siblings. Neither Florence nor Gregory, their parents, nor Gregory's grandmother have shown any signs of the disease. What is the risk that their child will inherit the fatal characteristic.

Figure 3. The Bayesian Network for the Genetic Reproduction Problem



Next, we construct a Bayesian network that models the inheritance of the fatal disease through the genealogical structure. In general, we are interested in determining the genotype of the individuals. To each individual, we associate a node labeled with his/her initial and having as states the possible genotypes identified by the combinations of alleles. Hence, each individual is of exactly one of the genotypes a_1a_1, a_1a_2, or a_2a_2.

In our problem, when allele a_1 is dominant, then this person is a carrier of the disease. This means that the genotypes a_1a_1 and a_1a_2 are carriers of the disease, whereas a_2a_2 is not.

What can be observed is whether the disease is present or not, but this can only be determined when the individual has reached a mature age due to the sneaky character of the disease. In order to be able to enter relevant information on observed occurrences and ask for expectations of the disease, we add a disease node to the elder and the upcoming generation. These nodes, labeled with the individuals initial with subscript d, have two possible states, “yes” and “no”, corresponding to the presence or the absence of the disease, respectively.

Table 3. Conditional Probability Tables for Genetic Reproduction Problem

α_A	$p(A = a_1a_1) = 0.0001$ $p(A = a_1a_2) = 0.0198$ $p(A = a_2a_2) = 0.9801$	α_B $p(B = a_1a_1) = 0.0025$ $p(B = a_1a_2) = 0.25$ $p(B = a_2a_2) = 0.7475$
α_{A_d}	$p(A_d = y \mid A = a_1a_1) = 1$ $p(A_d = y \mid A = a_1a_2) = 1$ $p(A_d = y \mid A = a_2a_2) = 0$	$p(A_d = n \mid A = a_1a_1) = 0$ $p(A_d = n \mid A = a_1a_2) = 0$ $p(A_d = n \mid A = a_2a_2) = 1$
α_E	$P(E = a_1a_1 \mid A = a_1a_1, B = a_1a_1) = 1$ $P(E = a_1a_2 \mid A = a_1a_1, B = a_1a_1) = 0$ $P(E = a_2a_2 \mid A = a_1a_1, B = a_1a_1) = 0$	$P(E = a_1a_1 \mid A = a_1a_2, B = a_2a_2) = 0$ $P(E = a_1a_2 \mid A = a_1a_2, B = a_2a_2) = 0.5$ $P(E = a_2a_2 \mid A = a_1a_2, B = a_2a_2) = 0.5$
	$P(E = a_1a_1 \mid A = a_1a_1, B = a_1a_2) = 0.5$ $P(E = a_1a_2 \mid A = a_1a_1, B = a_1a_2) = 0.5$ $P(E = a_2a_2 \mid A = a_1a_1, B = a_1a_2) = 0$	$P(E = a_1a_1 \mid A = a_2a_2, B = a_1a_1) = 0$ $P(E = a_1a_2 \mid A = a_2a_2, B = a_1a_1) = 1$ $P(E = a_2a_2 \mid A = a_2a_2, B = a_1a_1) = 0$
	$P(E = a_1a_1 \mid A = a_1a_1, B = a_2a_2) = 0$ $P(E = a_1a_2 \mid A = a_1a_1, B = a_2a_2) = 1$ $P(E = a_2a_2 \mid A = a_1a_1, B = a_2a_2) = 0$	$P(E = a_1a_1 \mid A = a_2a_2, B = a_1a_2) = 0$ P(E = a1a2 \mid A = a_2a_2, B = a_1a_2) = 0.5 $P(E = a_2a_2 \mid A = a_2a_2, B = a_1a_2) = 0.5$
	$P(E = a_1a_1 \mid A = a_1a_2, B = a_1a_1) = 0.5$ $P(E = a_1a_2 \mid A = a_1a_2, B = a_1a_1) = 0.5$ $P(E = a_2a_2 \mid A = a_1a_2, B = a_1a_1) = 0$	$P(E = a_1a_1 \mid A = a_2a_2, B = a_2a_2) = 0$ $P(E = a_1a_2 \mid A = a_2a_2, B = a_2a_2) = 0$ $P(E = a_2a_2 \mid A = a_2a_2, B = a_2a_2) = 1$
	$P(E = a_1a_1 \mid A = a_1a_2, B = a_1a_2) = 0.25$ $P(E = a_1a_2 \mid A = a_1a_2, B = a_1a_2) = 0.5$ $P(E = a_2a_2 \mid A = a_1a_2, B = a_1a_2) = 0.25$	

Probability assessments are given in Table 3. The interpretation of these probabilities is similarly to that for the previous problems. Our notation $A = a_1a_2$ stand for A has a genotype a_1a_2.

For nodes C and D we have the same probabilities as for node A, and the conditionals for these nodes are denoted by α_C and α_D, respectively. For nodes B_d, C_d, and H_d, we have the same probabilities as node A_d, and the conditionals for these nodes are denoted by α_{B_d}, α_{C_d}, and α_{H_d}, respectively. Nodes F, G, and H have two parents, and they have the same conditional probabilities as node E and these are labeled α_F, α_G, and α_H, respectively.

3 THE LAURITZEN-SPIEGELHALTER ARCHITECTURE

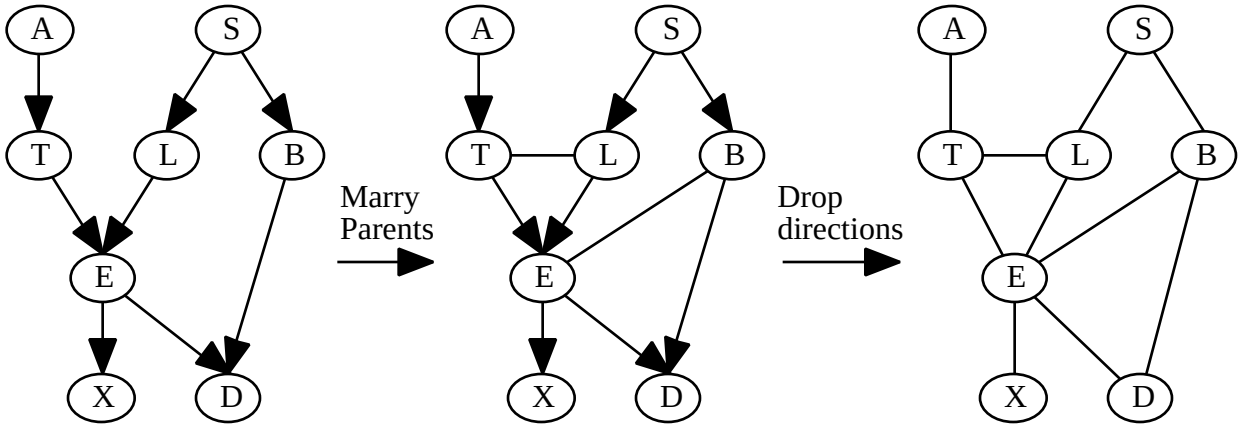
In this section , we describe the Lauritzen-Spiegelhalter architecture for computing marginals.

In a probabilistic model, we make inferences by computing the marginal of the joint probability distribution for the variables of interest. For simplicity, we will assume that we are interested in the marginal for all variables. When we have a large number of variables, computing the joint is computationally intractable. However, when the conditional potentials have small domains, we can compute the marginals of the joint without explicitly computing the joint.

In the LS architecture, first we construct a join tree called a junction tree, and then we propagate messages in the junction tree. The junction tree is constructed from the directed acyclic graph G as follows. First we construct a moral graph G^m, next we triangulate G^m, and finally we arrange the cliques in G^m in a join tree.

The procedure for transforming a Bayesian network G into a moral graph G^m is as follows. First we “marry parents” by adding undirected edges between every pair of parents, and then we drop directions, i.e., replace directed arcs by undirected edges (see Figure 4). The resulting undirected graph is called the *moral graph* G^m of G.

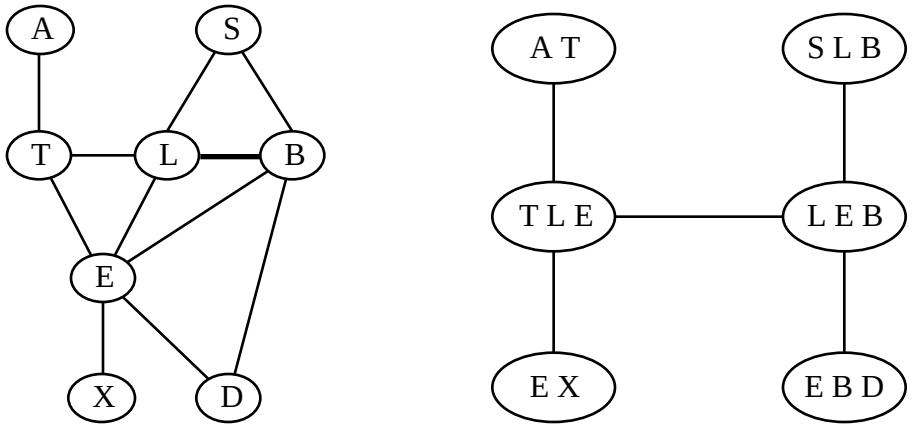
Figure 4. Constructing the Moral Graph G^m from the Bayesian network G



Next we triangulate the moral graph G^m if it is not a triangulated graph. An undirected graph is triangulated if every cycle of length $n \geq 4$ has a chord. Lauritzen and Spiegelhalter [1988] suggest the maximum cardinality search algorithm developed by Tarjan and Yannakakis [1984] for

checking whether an undirected graph is triangulated or not and for suggesting minimal fill-ins so that the resulting graph is triangulated. In the Chest Clinic problem, the moral graph shown in Figure 4 is not triangulated since we have a cycle SLEB of length 4 without a chord. A fill-in suggested by the maximum cardinality search algorithm is {L, B}. The resulting graph is triangulated (see Figure 5).

Figure 5 A Triangulated Graph and a Corresponding Junction Tree



Once we have a triangulated graph, we can arrange its cliques (maximal complete subsets of variables) in a join tree. A *join tree* is a tree whose nodes are subsets of variables such that if a variable is in two distinct nodes, then the variable must be in every node on the path between the two nodes. We will call the join tree whose nodes are the cliques of the triangulated moral graph a *junction tree*. This data structure enables local computations with potentials on domains within the cliques. A junction tree for the Chest Clinic problem is shown in Figure 5.

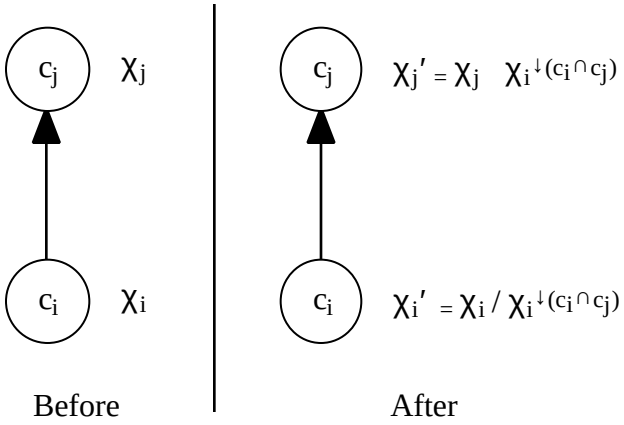
Next we associate each conditional potential κ_V with the clique that contains the subset {V} Pa(V). If we have observations, we model these as potentials and associate the potentials with a clique that includes the domain of the potential. If a clique has more than one potential associated with it, then we will assume that the combination of these potentials is associated with the clique.

For the Chest Clinic problem, suppose we have evidence that the patient has visited Asia and has Dyspnoea. We model this evidence as potentials o_A for {A}and o_D for {D}. It is easy to show that given the evidence, the posterior joint distribution is proportional to the product of all potentials including o_A and o_D.

Next we pick any node of the junction tree to be the root, and direct all edges of the junction tree toward the root. The propagation in Lauritzen and Spiegelhalter's architecture is done in two passes, inward and outward. In the inward pass, each node send a message to its inward neighbor, and in the outward pass, each node sends a message to its outward neighbors. Precise rules are as follows [Shafer 1996].

Inward Pass (see Figure 6)**:**

Figure 6. Inward Propagation (from c_i to c_j) in the LS Architecture

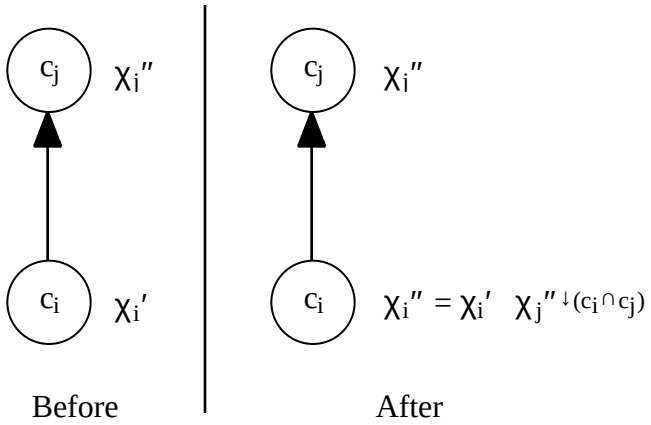


- **Rule 1**. Each node waits to send its message to its inward neighbor until it has received a message from all its outward neighbors. If a node has no outward neighbors, it can send a message right away.
- **Rule 2**. When a node is ready to send a message to its inward neighbor, it computes the message by marginalizing its current potential to its intersection with the inward neighbor. It sends this message to its inward neighbor, and then it divides its own current potential by the message.
- **Rule 3**. When a node receives a message from its outward neighbor, it replaces its current potential with the product of that potential and the message.

The inward pass ends when the root has received a message from all its outward neighbors.

Outward Pass (see Figure 7)**:**

Figure 7. Outward Propagation (from c_j to c_i) in the LS Architecture



- **Rule 1**. Each node waits to send its messages to its outward neighbors until it has received the message from its inward neighbor. The root which has no inward neighbor can send a message right away.
- **Rule 2**. When a node is ready to send a message to its outward neighbor, it computes the message by marginalizing its current potential to its intersection with the outward neighbor. It sends this message to its outward neighbor.
- **Rule 3**. When a node receives a message from its outward neighbor, it replaces its current potential with the product of that potential and the message.

The outward pass ends when all leaves have received messages from their inward neighbors. At the end of the outward pass, the potential associated with each clique is the marginal of the posterior joint for the clique (up to a normalization constant)..

Figures 8, 9 and 10 illustrate the computations in the LS architecture for the Chest Clinic problem.

Figure 8. At the Beginning

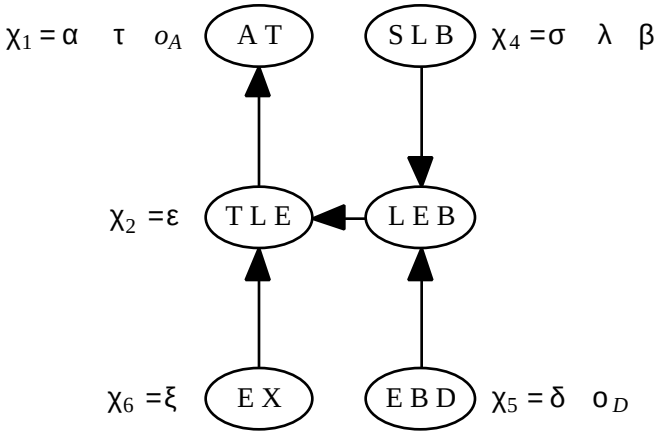


Figure 9. At the End of the Inward Propagation

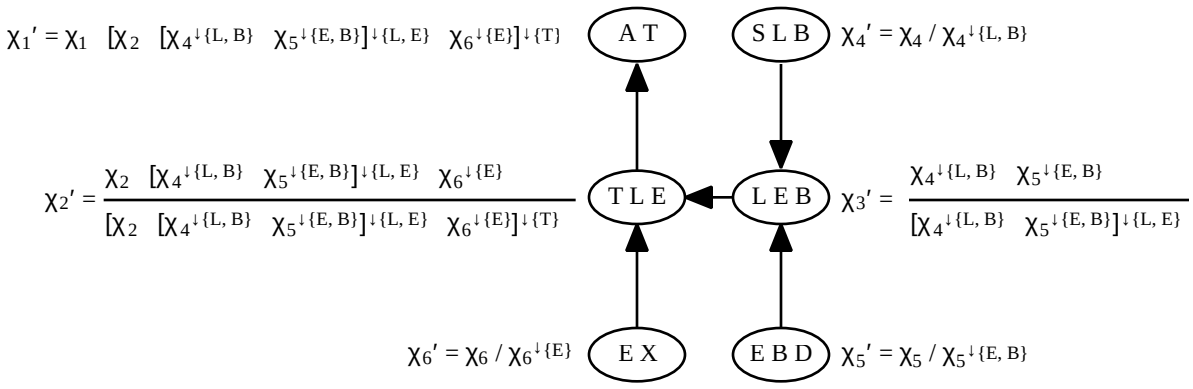
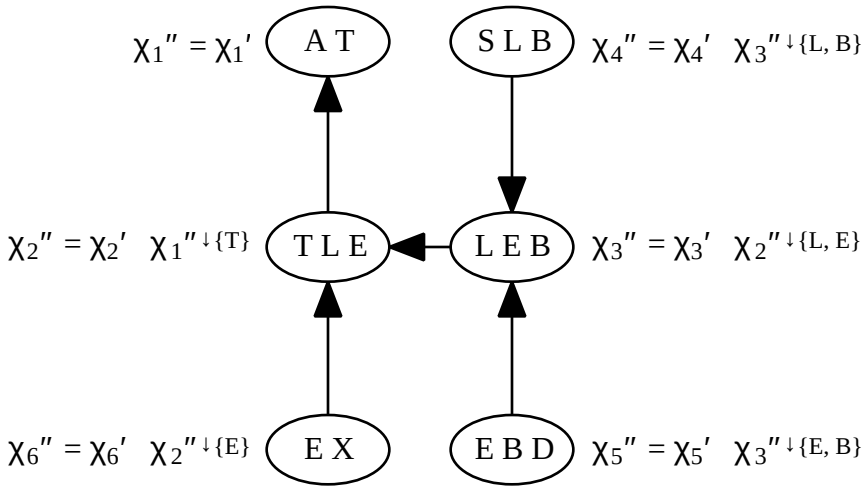


Figure 10. At the End of the Outward Propagation



At the end of the outward pass, we have the marginal of the posterior distribution at each clique. However, the stated task is the computation of the marginal of the posterior for each variable in the Bayes net. We can compute the marginal for a variable from any clique marginal that contains the variable. Since it is more efficient to compute the this marginal from a smaller clique, we will do so from a smallest clique that contains the variable. For example, to compute the marginal for E in the Chest Clinic problem, we can do so from the marginals of the following cliques: {T, L, E}, {L, E, B}, {E, B, D} and {E, X}. Since {E, X} is the clique with the smallest number of states, it is most efficient to compute the marginal for E from {E, X}. Of course, this strategy ignores the computational cost of identifying a smallest clique.

4 THE HUGIN ARCHITECTURE

In this section, we sketch the Hugin architecture. Although it was initially described for computing marginals of probability distributions [Jensen *et al*. 1990a, b], it has been recently extended by Lauritzen and Jensen [1996] so that it is more widely applicable to domains that satisfy some axioms.

We start by assuming that we have a junction tree and the corresponding probability potentials for each clique. We introduce a storage register between every two cliques whose domain is the intersection of the two cliques. We call this storage register a *separator*. Pick any node to be the root. The propagation in Hugin architecture is done in two passes, inward and outward. In the inward pass, each node send a message to its inward neighbor, and in the outward pass, each node sends a message to its outward neighbors.

In the Hugin architecture, in the inward pass the sender does not divide the message. Instead, we save it in the separator. This requires more space, but it save computations (as we will see shortly). On the outward pass, the separator divides the outward message by the message it has stored before passing it on to be multiplied into the potential of the receiving node. Notice that the division is done in the separator which has a smaller state space than either of the two cliques.

If we assume that at the beginning, each separator has the corresponding identity potential ι (a potential whose values are identically one, and whose domain is same as the separator), then the inward action is same as the outward. Precise rules are as follows [Shafer 1996]:

Figure 11. Inward Propagation (from c_i to c_j) in the Hugin Architecture

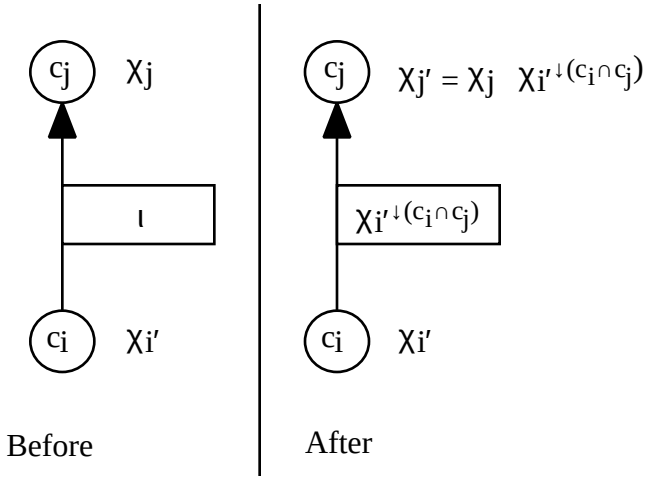
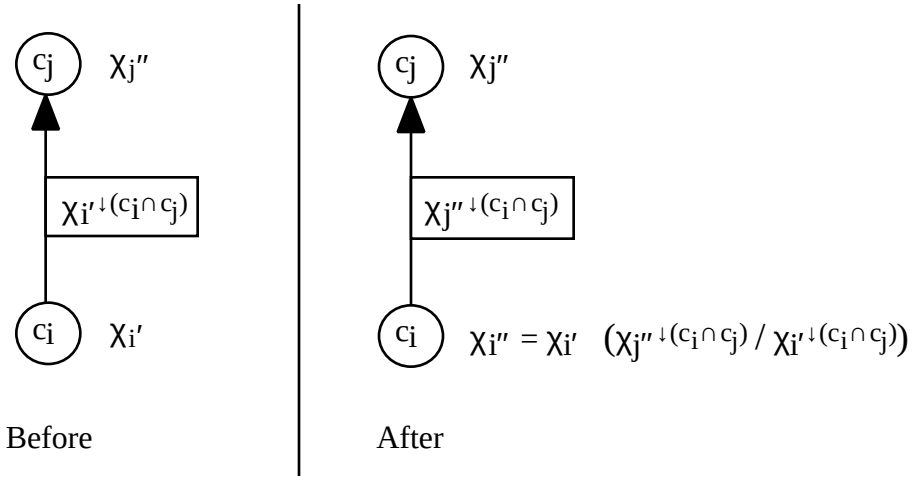


Figure 12. Outward Propagation (from c_j to c_i) in the Hugin Architecture



- **Rule 1**. Each non-root node waits to send its message to a given neighbor until it has received messages from all its other neighbors.
- **Rule 2**. The root waits to send messages to its neighbors until it has received messages from them all.
- **Rule 3**. When a node is ready to send its message to a particular neighbor, it computes the message by marginalizing its current potential to its intersection with this neighbor, and then it sends the message to the separator between it and the neighbor.
- **Rule 4**. When a separator receives a message *New* from one of its two nodes, it divides the message by its current potential *Old*, send the quotient *New*/*Old* on to the other node, and then replaces *Old* with *New*.
- **Rule 5**. When a node receives a message, it replaces its current potential with the product of the potential and the message.

Rules 1 and 2 force the propagation to move in to a "root" and then back out.

At the end of the propagation, the potentials on all the nodes and separators are marginals of the posterior joint P χi.

Suppose Θ is the set of all cliques, and Γ is the set of all separators. Then at the beginning, at the end of the inward pass, at the end of the outward pass, or at any step in the propagation process, P (i Θχi) / (i Γ χi).

Figures 13, 14 and 15 illustrate the computations in the Hugin architecture for the Chest Clinic problem.

Figure 13. At the Beginning

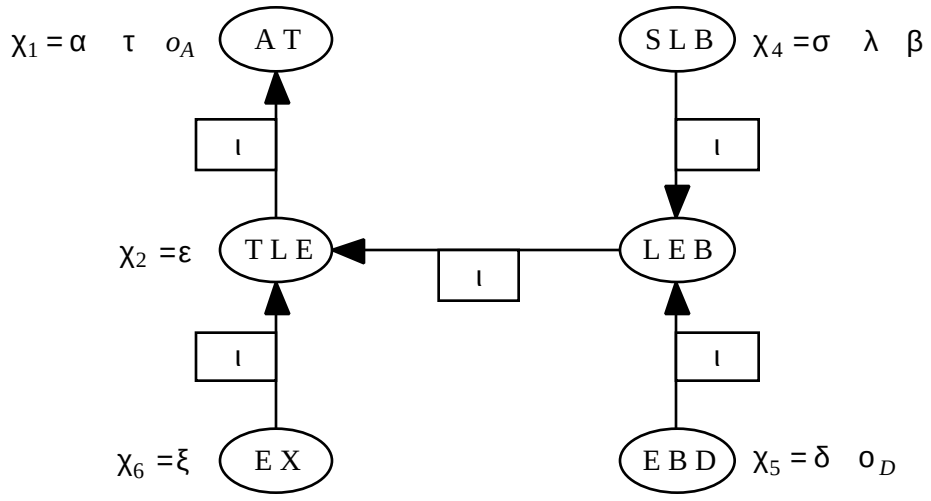


Figure 14. At the End of the Inward Propagation

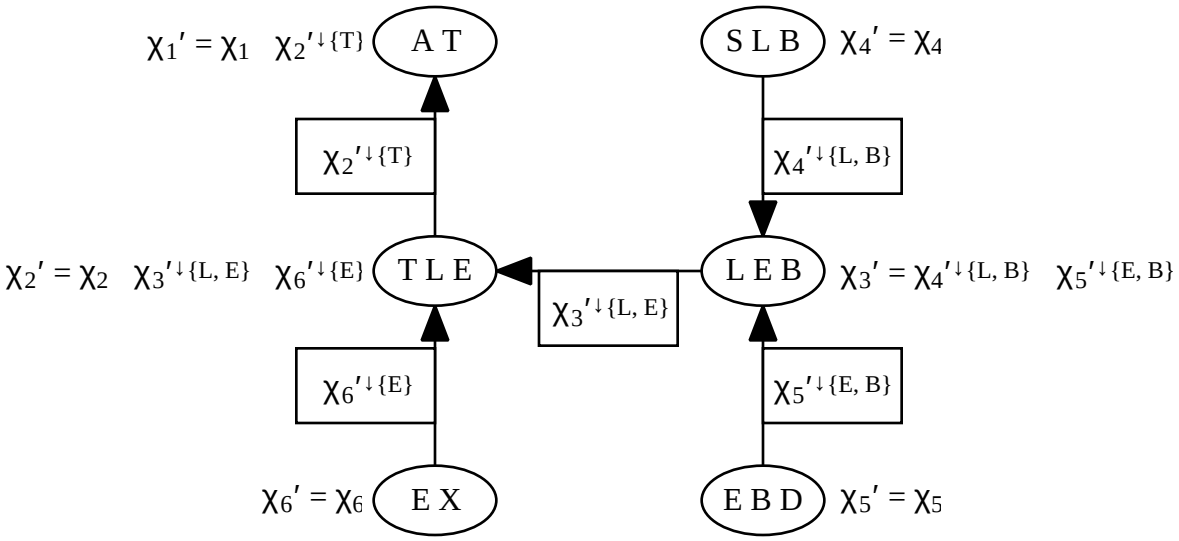
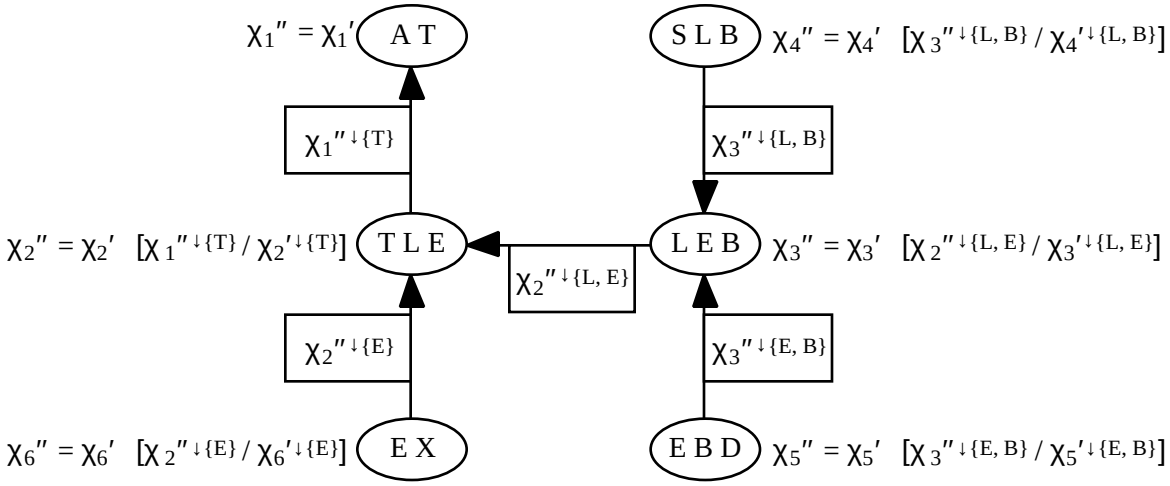


Figure 15. At the End of the Outward Propagation



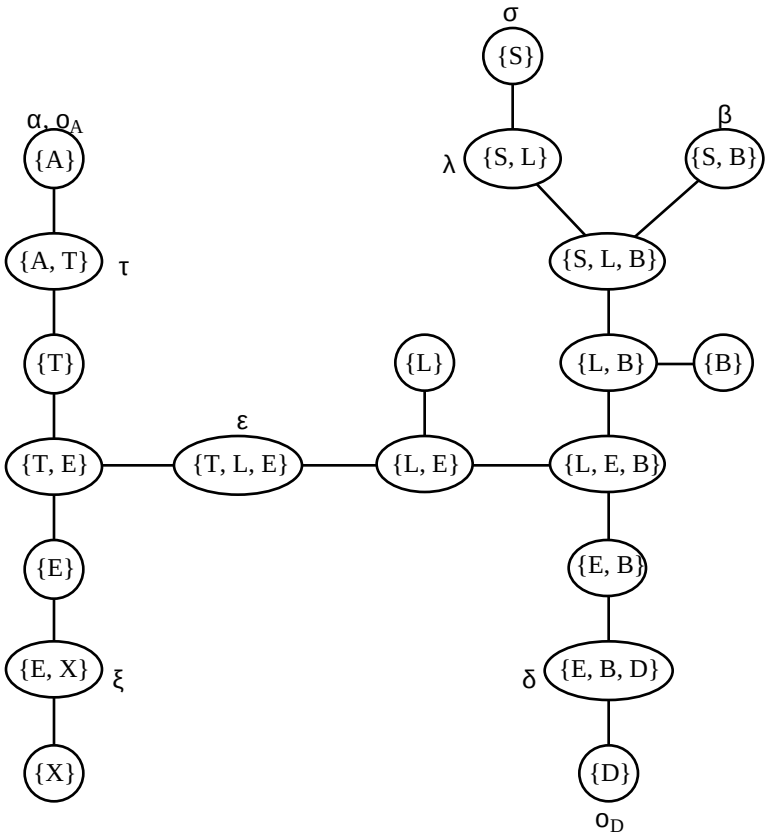
We compute the marginal for a variable from the marginal for a smallest separator that contains the variable. If there is no separator that contains the variable then we compute it from the marginal for a smallest clique that contains the variable. Like in the LS architecture, this strategy ignores the computational cost of identifying the smallest separator or clique that contains the variable.

5 THE SHENOY-SHAFER ARCHITECTURE

In this section, we sketch the Shenoy-Shafer architecture and illustrate it using the Chest Clinic problem.

In the Shenoy-Shafer architecture, we start with a collection of potentials that define the joint distribution. The domains of the potentials form a hypergraph. To this hypergraph, we add subsets for which we desire marginals. For example, if we wish to compute marginals for each singleton variable, we add these singleton variables to the hypergraph if they are not already included in the hypergraph. In the Chest Clinic problem, we start with a set of potentials $\vartheta = \{\alpha, \sigma, \tau, \lambda, \beta, \varepsilon, \xi, \delta, o_A, o_D\}$, and a hypergraph $H = \{\{A\}, \{S\}, \{A, T\}, \{S, L\}, \{S, B\}, \{T, L, E\}, \{E, X\}, \{E, B, D\}, \{D\}, \{T\}, \{L\}, \{B\}, \{E\}, \{X\}\}$.

Figure 16. A Binary Join Tree for the Chest Clinic Problem.

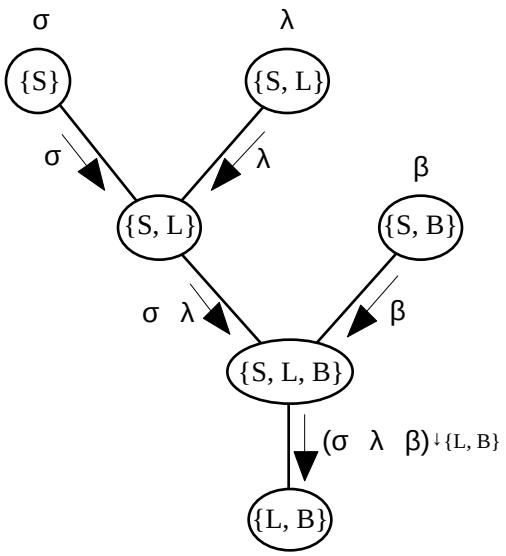


The first step in the Shenoy-Shafer architecture is to arrange the subsets in H in a binary join tree. A binary join tree is a join tree such that no node has more than three neighbors. The binary join tree construction process is motivated by the idea of fusion [Shenoy 1992] (called peeling by Cannings *et* al. [1968]), and the idea that all combinations should be done on a binary basis, i.e., potentials should be multiplied two at a time. A binary join tree is a data structure designed to cache

computation so as to reduce the computation involved in combination and marginalization. A binary join tree for the hypergraph in the Chest Clinic problem is shown in Figure 16.

Shenoy [1997] describes a formal procedure for constructing a binary join tree. Here we will sketch this procedure. As per the fusion algorithm, when we delete a variable, we combine all potentials that contain the variable in their domains and then marginalize the variable out of the combination. The potentials that do not contain the variable in their domains remain unchanged. For example, in the Chest Clinic problem, if we fuse the potentials in ϑ with respect to S, we get $\{(\sigma \otimes \lambda \otimes \beta)^{\downarrow \{L, B\}}, \alpha, \tau, \varepsilon, \xi, \delta, o_A, o_D\}$. The computation of $\sigma \otimes \lambda \otimes \beta$ is achieved using binary fusion, i.e., we first combine σ and λ, and then we combine $\sigma \otimes \lambda$ and β. This suggests the binary subtree shown in Figure 17. If we recursively implement this using the deletion sequence, say, XASDBLE, we get the binary join tree shown in Figure 18.

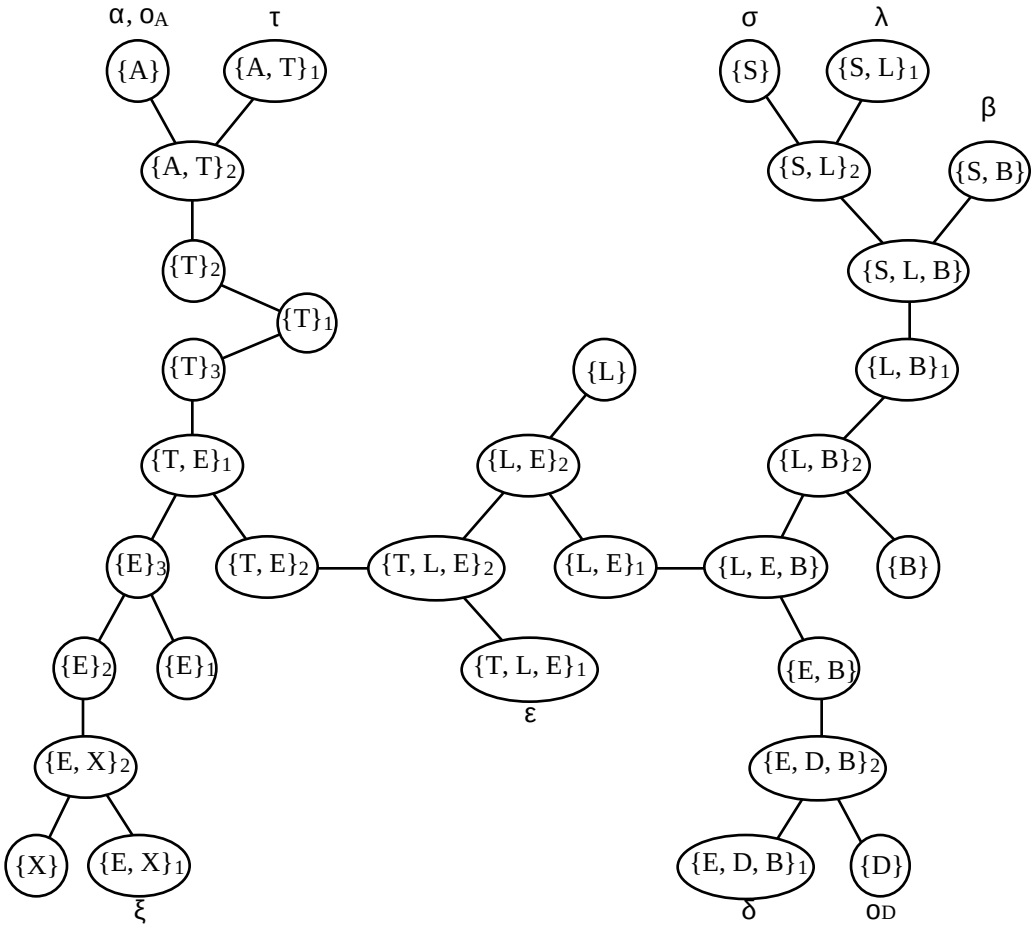
Figure 17. A Binary Join Tree Suggested by Binary Fusion with respect to S



Notice that in Figure 18, many nodes are duplicated. If we have a pair of duplicate nodes that are neighbors and merging these two nodes does not increase the number of neighbors of the merged node to more than three, then we can merge the duplicated nodes into one node. If we do this in the Chest Clinic problem, the condensed binary join tree that is obtained is the one shown in Figure 16. In general, we may not be able to always get rid of duplicate nodes [Shenoy 1997].

Once we have a binary join tree, we associate each potential with one of the subsets in the binary join tree that corresponds to its domain. Next, each node in the tree that needs to compute the marginal for it requests a message from each of its neighbors. The messages are computed using Rule 1 as follows.

Figure 18. A Binary Join Tree for the Chest Clinic Problem Suggested by Binary Fusion



Rule 1 (*Computing Messages*) Suppose *r* and *s* are neighbors, and suppose *s* has requested a message from *r*. *r* in turn requests messages from its other neighbors, and after it has received these messages, it computes the message to *s* as follows. Informally, the message that node *r* sends to its neighbor *s* is the combination of all messages that *r* receives from its other neighbors together with its own probability potential marginalized to $r \cap s$. Formally, suppose $\mu^{r \to s}$ denotes the message from *r* to *s*, suppose N(*r*) denotes the neighbors of *r* in the binary join tree, and suppose α_r denotes the probability potential associated with node *r*. Then the message from node *r* to its neighboring node *s* is computed as follows:

$$\mu^{r \to s} = \Big(\big\{ \mu^{t \to r} \mid t \in (\mathrm{N}(r) - \{s\}) \big\} \cup \alpha_r \Big)^{\downarrow r \cap s}$$

Notice that a leaf of the join tree has only one neighbor and therefore when it has received a request for a message, it can send it right away without waiting for any messages.

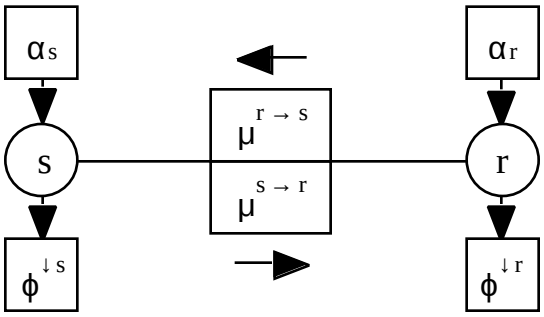
In Figure 17, notice that the messages displayed there satisfy Rule 1 above. When a node that needs to compute the marginal for it has requested and received messages from all its neighbors, then it computes the desired marginal using Rule 2 as follows.

Rule 2 (*Computing Marginals*) When a node r has received a message from each of its neighbors, it combines all messages together with its own probability potential and reports the results as its marginal. If ϕ denotes the joint potential, then

$$\phi^{\downarrow r} = \{\mu^{t \to r} \mid t \quad N(r)\} \quad \alpha_r$$

Each node in the binary join tree will have zero, one, two or more storage registers, one for each input probability potential (if any), and one for reporting the marginal of the joint (if a marginal for the node is desired). Each edge (separator) in the join tree would have at most two storage register for the two messages, one in each direction. Figure 19 shows the storage architecture for a simple join tree with two nodes. Each of the two nodes is assumed to have one input potential. Also, we assume that we desire the marginal for both nodes. Notice that the domain of the separator between r and s is $r \cap s$.

Figure 19. The Shenoy-Shafer Architecture for a Join Tree with Two Nodes



In the Chest Clinic problem, suppose we desire marginals for each of the variables in the problem. To achieve this, suppose that the singleton nodes {A}, {S}, {T}, {L}, {B}, {E}, {X}, and {D} in the binary join tree of Figure 16 request a message from their neighbors. Notice that not all messages are computed. For example, the message $\mu^{SLB \to SB}$ is not computed since it is not requested by any node.

Notice that unlike the LS and Hugin architectures, there are no division operations in the SS architecture. Also, notice that unlike the LS and Hugin architectures, the input potentials remain unchanged during the propagation process in the SS architecture. Notice also that the marginal of the joint potential for a variable is computed at the corresponding singleton variable node of the binary join tree.

6 COMPARISON

In this section, we will compare the Lauritzen-Spiegelhalter (LS), Hugin, and Shenoy-Shafer (SS) architectures. In the comparison, we will focus our attention on the graphical structure for message

propagation, the message-passing scheme, the storage efficiency, and the computational efficiency of each architecture.

In all three architectures, we assume that we start with a Bayesian network representation of a problem and that we have some evidence (observations or likelihoods) for some variables. The task is to compute the marginals of the posterior distribution for all variables in the problem.

Graphical Structures for Message Propagation. In the LS and Hugin architectures, propagation of potentials is done in a junction tree. In the SS architecture, propagation of potentials is done in a binary join tree. The nodes of a junction tree are the cliques of a triangulated moral graph of the original Bayesian network. A corresponding binary join tree includes these cliques as well as several subsets of these cliques. Therefore, a binary join tree has more nodes than in a corresponding junction tree. For example, in the Chest Clinic problem, the junction tree shown in Figure 5 has six nodes whereas the corresponding binary join tree shown in Figure 16 has 20 nodes. For the Stud Farm problem, the junction tree shown in Figure A.1 has 9 nodes and the corresponding binary join tree in Figure A.2 has 34 nodes. And in the Genetic Reproduction problem, the junction tree shown in Figure A.3 has 10 nodes and the corresponding binary join tree shown in Figure A.4 has 28 nodes. (Notice that if we start with a binary join tree and we condense it by absorbing adjacent nodes that are subsets/supersets of each other, we get a "corresponding junction tree.")

The junction tree yields only marginals for the cliques in the LS architecture, and marginals for cliques and separators in the Hugin architecture. Since our stated task is to compute marginals of singleton variables, there is further computation needed in these two architectures. In the LS architecture, the marginal for a variable can be computed most efficiently from the marginal of the smallest clique containing the variable. However, identifying the smallest clique itself involves some computation. In the Hugin architecture, if a variable belongs to a separator, then the marginal for the variable can be computed most efficiently from a smallest separator containing the variable. If a variable does not belong to any separator, then its marginal can be computed most efficiently from a smallest clique containing the variable. Identifying whether a variable is in some separator or not, and identifying a smallest separator or a smallest clique containing the variable involves some computation. In the SS architecture, if during the construction of a binary join tree, we include all singleton subsets, then the graphical structure yields marginals for singletons at the end of the message passing stage with no further computation required.

It is not necessary that we use junction trees for the LS and Hugin architectures. We could use any join tree including binary join trees. However, given the message passing schemes of these two architectures, it is inefficient (with respect to both computation and storage) to implement these two message passing schemes on join trees with many nodes. We will be more specific about this aspect when we discuss computational efficiencies of the three architectures. Also, it is not

necessary that we use a binary join tree for the SS architecture. We could use any join tree including junction trees. However, there is computational penalty in using non-binary join trees or condensed junction trees for the SS message passing scheme. For these reasons, the LS architecture is associated with junction trees, the Hugin architecture is associated with junction tree with separators, and the SS architecture is associated with binary join trees constructed in the manner described in Shenoy [1997].

Message-Passing Schemes. In the LS architecture, first we arbitrarily designate a clique of the junction tree as the root. The propagation of messages in done in two stages—the inward phase where each clique send a message to its inward neighbor, and the outward phase in which each clique sends a message to each of its outward neighbors. At the beginning we have an evidence potential representation. And at the end of the outward phase, at each clique, we have the marginals for it. Each clique in the junction tree stores a potential. Computations are done by each clique in the junction tree.

In the Hugin architecture, we designate a node as the root. Each clique send a message to each of the separators between it and its neighbors. When a separator receives a message from one of its neighboring clique, it sends a message to its other neighboring clique. At all times, the joint potential is equal to the product of the potentials at the cliques divided by the product of the potentials at the separators. When all messages have been sent, the potential at each clique and at each separator is the marginal of the joint for that node. Each clique and each separator in the junction tree stores a potential. Computations are done by each clique and by each separator in the junction tree.

In the SS architecture, nodes for which the marginals are desired request messages from all their neighbors. When a node receives a request for a message, it in turn requests messages from all its other neighbors. When all requested messages have been delivered, the marginals are computed at the desired nodes. A node may store either no potential, or one potential (input or output) or two or more potentials (one for each input, and output). Each edge (separator) between two nodes may store one or two potentials. Computations are done only by nodes and not by separators.

Although we have restricted our study in this article to Bayesian networks, all three architectures are applicable more widely. Lauritzen and Jensen (1996) have described axioms that generalize the LS and the Hugin architecture to other domains. These axioms include the axioms proposed by Shenoy and Shafer (1990). A natural question is how generally applicable are these three architectures. Since the Shenoy-Shafer architecture does not use the division operation, it is clear that the Shenoy-Shafer architecture is more widely applicable that the Lauritzen-Spiegelhalter or the Hugin architecture. For example, the problem of fast retraction proposed by Cowell and David [1992] can be handled by all three architectures in the probabilistic domain. However, fast

retraction cannot be handled in non-probabilistic domains by the Lauritzen-Spiegelhalter and Hugin architectures as the axioms are not satisfied [Lauritzen and Jensen 1996]. Fast retraction is easily handled in the Shenoy-Shafer architecture [Lauritzen and Shenoy 1996].

Storage Efficiencies. In the LS architecture, each clique in the junction tree stores one potential. Thus the total storage requirements will depend on the number of cliques in the junction tree and state spaces of the cliques. If after propagating the messages in the junction trees, we get a new piece of evidence, then we will have to start again with the input and evidence potentials. Also, a user may want to edit the input and evidence potentials. For these two reasons, we have to also include the storage requirements for the input and evidence potentials. Also, at the end of the outward propagation, we have only the marginals for the cliques. However, our stated task is the computation of the marginals for each variable. These marginals are computed from the clique marginals. We will also include the storage requirements for storing the marginals of each variable.

In the Hugin architecture, each clique in the junction tree stores one potential. Also, each separator between two adjacent cliques stores one potential. Also, a user may need to edit the input and evidence potentials. So these need to be stored separately. Therefore, we will also include the storage space for storing the input and evidence potentials. Also, when all messages have been computed, we have only the marginals for the cliques and separators. We still need to compute marginals of singleton variables. So we will include storage space for the marginals of each variable.

In the SS architecture, each node may have either zero, one, two or more potentials. If a node has at least one input potential and it is a singleton node whose marginal is desired, then such a node with have two or more potentials. If a node has neither an input potential nor is the marginal for the node desired, then it will have zero potentials. In all other cases, it will have one potential (either an input potential or an output potential). If we regard the edge between two adjacent nodes as a separator, then each separator will have either one or two potentials depending on which messages are requested. If both adjacent nodes request messages from each other, then each separator will store two potentials. If only one message is requested, then a separator will store only one potential.

Table 4. Storage Efficiencies of the Three Architectures for Three Sample Problems

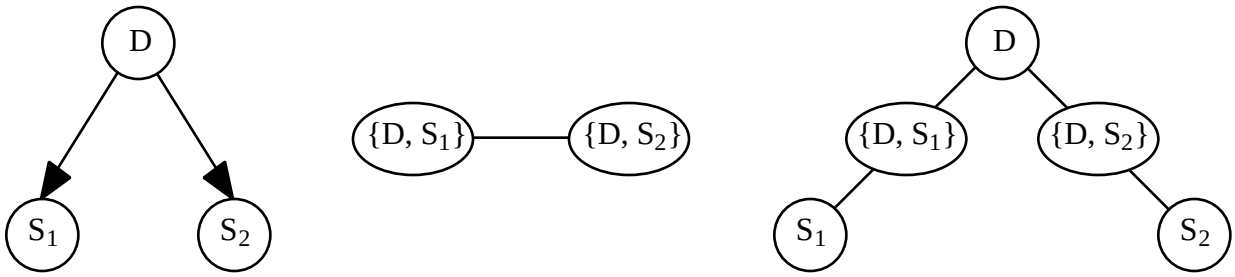
Storage Efficiency	Architectures		
# floating point numbers (fpn)	LS	Hugin	SS
Chest Clinic with evidence for A and D	96	112	158
Stud Farm	214	262	376
Genetic Reproduction	368	425	457

Table 4 displays the storage requirements for the three problems described in Section 2. In this table, the storage requirements are described in units of floating point numbers (fpn). Thus, e.g., to store a potential whose domain consists of three binary variables, we will need storage space of $2^3 = 8$ fpn.

In general, it is easy to see that, assuming we are working with the same junction tree, the Hugin architecture will have always more storage requirements than the LS architecture because of storage at the separators.

In comparing the storage requirements of Hugin with SS architectures, there are no general results. Although a binary join tree has more nodes than a corresponding junction tree, not every node in a binary join tree has a potential associated with it. All input and evidence potential are included in both architectures and all output potentials are also included in both architectures. So the differences in storage are due to storage at cliques and separators in the Hugin architecture and storage at separators in the SS architecture. In the Hugin architecture, all separators include exactly one potential each, whereas in the SS architecture, most separators include two potentials and there are usually a lot more separators in a binary join trees than in corresponding junction trees. However, every clique in a junction tree stores a potential whereas these potentials are not present in the SS architecture.

Figure 20. A Bayes net, a Junction Tree, and a Binary Join Tree



In Table 4, we see that the SS architecture has more storage than the LS and Hugin architectures for the three problems. It is easy to construct an artificial problem in which the SS architecture has less storage than the LS and Hugin architectures. Consider a Bayes net with one disease variable D and two symptom variables S_1 and S_2 as shown in Figure 20. Suppose we have two pieces of evidence for nodes S_1 and S_2, respectively. A junction tree and a binary join tree are also shown in Figure 20. Suppose that each of the three variable has 5 states. Then in all three architectures we have the same storage for input (5 + 25 + 25 = 55 fpn), evidence (5 + 5 = 10 fpn) and output potentials (3*5 = 15 fpn). In the LS architecture we have a storage of 50 (= 2*25) fpn at the two cliques in the junction tree. In the Hugin architecture, we have a total storage of 55 (= 2*25 + 5) fpn at the two cliques and one separator. In the SS architecture, we have a total storage

of 40 (= 4*2*5) at the 4 separators. Thus in this problem, SS has less storage than both the LS and Hugin architectures.

Computational Efficiency. It is traditional to study worst case order of magnitude complexity of computational algorithms. From this perspective, there are no essential differences between the three architectures. All three architectures compute the marginals using local computation. In the worst case, the computational complexity of the three algorithms are exponential in the size (# variables) of the largest clique.

Table 5. # Binary Arithmetic Operations for Some Sample Problems

# Binary Arithmetic Operations	Architecture		
Problem	LS	Hugin	SS
Chest Clinic with no evidence			
# binary additions	72	60	56
# binary multiplications	84	84	122
# binary divisions	36	16	
Chest Clinic with evidence for A and D			
# binary additions	72	60	56
# binary multiplications	96	96	124
# binary divisions	36	16	
Chest Clinic with evidence for A, D, S and X			
# binary additions	72	60	56
# binary multiplications	108	108	126
# binary divisions	36	16	
Stud Farm with evidence for J			
# binary additions	221	179	187
# binary multiplications	248	248	345
# binary divisions	108	48	
Genetic Reproduction with evidence for A_d, B_d and C_d			
# binary additions	400	346	334
# binary multiplications	411	411	522
# binary divisions	159	57	

Here we will look at computational efficiencies of the three architectures using a very crude measure: # binary arithmetic operations (additions, multiplications, and divisions). It is clear that this crude measure does not describe the actual computational efficiency. This measure does not include other operations such as table lookups, comparisons, read/write to memory, etc. Even this

crude measure is difficult to measure in general. Our methodology is as follows. We solve the three problems described in Section 2 under several scenarios, and we count the number of binary additions, multiplications and divisions. Based on these observations, we identify the sources of relative inefficiencies in each of the three architectures. And we list as many general conclusions as we can.

First, the Hugin architecture always does fewer additions than the LS architecture. This is because computation of marginals of singleton variables is always done from clique marginals in the LS architecture whereas in the Hugin architecture, it is done from the separator marginals for some variables and clique marginals for some variables. Notice that we are ignoring the computational cost of identifying a smallest clique in the LS architecture and the cost of finding a smallest separator or clique in the Hugin architecture.

Second, the LS and Hugin architectures always do the same number of multiplications. The Hugin architecture is an adaptation of the LS architecture, and it is not surprising that this aspect of the two architectures is the same.

Third, the Hugin architecture always does fewer divisions than the LS architecture. The Hugin architecture does divisions in the separator whereas the LS architecture does divisions in the cliques. This was a major motivation that led to the Hugin architecture. Since the Hugin architecture is more computationally efficient than the LS architecture, we will restrict our comparison of the SS architecture to the Hugin architecture.

Comparing the Hugin and SS architectures, in Table 5, we notice that sometimes SS does fewer additions than Hugin (e.g., in Chest Clinic and Genetic Reproduction problems) and sometimes Hugin does fewer additions than SS (e.g., in Stud Farm problem). A detailed examination of the addition operations done in the two architectures reveals the following reasons.

In the Chest Clinic problem, SS does 4 fewer additions than the Hugin architecture. During the outward propagation of the Hugin architecture (see Figure 15), node TLE computes two potentials, $\chi''^{\downarrow\{L, E\}}$ (stored at separator {L, E}) and $\chi''^{\downarrow\{E\}}$ (stored at separator {E}). Computation of $\chi''^{\downarrow\{L,\ E\}}$ from χ'' requires 4 additions, and computation of $\chi''^{\downarrow\{E\}}$ from χ'' requires 6 additions for a total of 10 additions. If we had computed $\chi''^{\downarrow\{E\}}$ from $\chi''^{\downarrow\{L, E\}}$, we would have done only 2 additions thus saving 4 additions. But there is no way we can do this given the arrangements of cliques in the junction tree. In the SS architecture for the Chest Clinic problem, the binary join tree (shown in Figure 16) has more nodes than in the junction tree and consequently there is more caching of messages than in the Hugin architecture. The marginal for E is computed from the node {T, E} and this requires only 2 additions.

In the Stud Farm problem, Hugin does 8 fewer additions than the SS architecture. Hugin computes the marginal of B from {B, E} (the smallest separator containing B), and computes the marginal of H from {H, I}, the smallest separator containing H (see Figure A.1 in the Appendix).

Computing these two marginals requires 4 additions. In the SS architecture, the binary join tree is constructed from the viewpoint of minimizing multiplications. Thus, the singleton node {E} is connected to {A, E, H} and the singleton node {H} is connected to {A, E, H} via {A, H} (see Figure A.2 in the Appendix). Thus computing marginal of E from {A, E, H} requires 6 additions and computing marginal of H from {A, E, H} via {A, H} requires 6 additions for a total of 12 additions, 8 more than in the Hugin architecture.

The number of multiplications done in Hugin and SS cannot be compared without accounting the division operations in the Hugin architecture. In a sense, the Hugin does division operations to avoid some multiplications (done in the SS architecture). Therefore we need to compare the aggregate of multiplications and divisions in Hugin to multiplications in the SS architecture. To aggregate the number of multiplications and divisions, we can simply add them. Alternatively, since on most chip architectures, a floating point division takes roughly 30 percent more time than doing a floating point multiplication, we can aggregate multiplications and divisions by assuming 1 ÷ = 1.3 ×.

For some problems, Hugin does fewer multiplications and divisions than the SS architecture, and for some problems, SS does fewer multiplications than the aggregate of multiplications and divisions in Hugin. A detailed examination of the multiplications and divisions done in the two architectures reveals the following reasons.

Hugin does divisions as a substitute for multiplications (in the SS architecture), but it does these divisions in the separators instead of in the cliques. On the other hand, the SS architecture avoids divisions by doing multiplications in the cliques. For example, in the Hugin architecture for the Chest Clinic problem, clique {L, E, B} does 8 multiplications in the inward stage, 8 multiplications in the outward stage, and 4 divisions in the separator {L, E} for a total of 16 multiplications and 4 divisions. On the other hand, in the SS architecture, {L, E, B} does 8 multiplications in the inward stage, and 16 multiplications in the outward stage (to send messages to {L, B} and {E, B}) for a total of 24 multiplications. Thus, even if we count a division as 1.3 multiplication, Hugin is more efficient than SS.

In the SS architecture for the Chest Clinic problem, first we multiply σ and λ at node {S, L} that requires 4 multiplications, and then we multiply $\sigma \quad \lambda$ and β at node {S, L, B} requiring 8 multiplications for a total of 12 multiplications. In the Hugin architecture for the Chest Clinic problem, at the outset, we multiply σ, λ and β at clique {S, L, B} for a total of 16 multiplications, 4 more than the SS architecture. Also, consider the multiplications done by clique {T, L, E} during the inward stage—$\chi_2 \quad \chi_3'^{\downarrow \{L, E\}} \quad \chi_6'^{\downarrow \{E\}}$. This requires 16 multiplications. Had we done these on a binary basis by multiplying $\chi_3'^{\downarrow \{L, E\}}$ and $\chi_6'^{\downarrow \{E\}}$ on {L, E} and then χ_2 and $\chi_3'^{\downarrow \{L, E\}} \quad \chi_6'^{\downarrow \{E\}}$ on {T, L, E}, we would have done 12 multiplications. Since there is no guarantee of doing binary

multiplications in a junction tree, the Hugin architecture does more multiplications than is done by the SS architecture in a join tree crafted to guarantee binary multiplications.

Notice that the Hugin propagation can be done in any join tree assuming we start with a clique marginal representation of the joint probability distribution. However since computations are done at each node and at each separator, there is a computational penalty in introducing additional nodes and separators. For example for the Chest Clinic problem with evidence for A and D, if we do Hugin propagation in the binary join tree shown in Figure 16, it requires 56 additions, 170 multiplications and 52 divisions (see Table A.10 in the Appendix) compared to 60 additions, 96 multiplications and 16 divisions for the junction tree of Figure 5 (see Table A.7 in the Appendix). Clearly, for the Hugin architecture, the junction tree in Figure 5 is more efficient than the binary join tree of Figure 16.

The SS propagation can be done in any join tree assuming we start with an evidence potential representation of the joint probability distribution. Since no computations are done at any separator, and since the computations done at a node depends on the number of neighbors, there may be a computational penalty if we use arbitrary join trees. For example, for the Chest Clinic problem with evidence for A and D, if we do SS propagation in the junction tree shown in Figure 5, it requires 60 additions, and 140 multiplications (see Table A.9 in the Appendix) compared to 56 additions and 124 multiplications for the binary join tree of Figure 16 (see Table A.8 in the Appendix). Clearly, for the SS architecture, the binary join tree in Figure 16 is more efficient than the junction tree of Figure 5 even though the latter is a binary join tree. Notice that if we restrict ourselves to cliques, there is no guarantee that we can always find a junction tree that is a binary join tree.

Overall, comparing LS and Hugin architectures, Hugin is computational more efficient than LS, whereas LS is more storage efficient than LS. In a sense, Hugin sacrifices storage efficiency to achieve better computational efficiency. We cannot make any general statements regarding relative storage or relative computational efficiencies of Hugin and SS architectures. There are some aspects of the Hugin architecture that are better than the SS architectures, namely division in separators. There are some aspects of the SS architecture that are better than the Hugin architecture, namely binary multiplications. Whether we can improve on these two architectures by borrowing the strengths of the other is a topic that needs further research.

Table 6. Computational Efficiency of the Three Architectures for Some Sample Problems

Computational Efficiency Total # unit binary operations (ubo)	Architecture LS	 Hugin	 SS
Assuming 1÷ = 1 × = 1 + = 1 ubo			
Chest Clinic with no evidence	192	160	178
Chest Clinic with evidence for A and D	204	172	180
Chest Clinic with evidence for A, D, S and X	216	184	182
Stud Farm with evidence for J	577	475	532
Genetic Reproduction with evidence for A_d, B_d and C_d	970	814	856
Assuming 1 + = 1 × = 1 ubo, 1÷ = 1.3 ubo			
Chest Clinic with no evidence	202.8	164.8	178
Chest Clinic with evidence for A and D	214.8	176.8	180
Chest Clinic with evidence for A, D, S and X	226.8	188.8	182
Stud Farm with evidence for J	609.4	489.4	532
Genetic Reproduction with evidence for A_d, B_d and C_d	1017.7	831.1	856

ACKNOWLEDGMENTS

This research was initiated during Spring 1996 when the second author was visiting the Institute of Informatics at the University of Fribourg. The authors are grateful for support and encouragement from Professor Juerg Kohlas. The paper has benefited from comments and suggestions by Bernard Anrig, Rolf Haenni, Tuija Isotalo, Juerg Kohlas, Norbert Lehmann, Paul-Andre Monney, and Dennis Nilsson.

REFERENCES

1. Cannings, C., E. A. Thompson and M. H. Skolnick (1978), "Probability functions on complex pedigrees," *Advances in Applied Probability*, **10**, 26-61.
2. Cowell, R. and A. P. Dawid (1992), "Fast retraction of evidence in a probabilistic expert system," *Statistics and Computing*, **2**, 37–40.
3. Jensen, F. V. (1996), *An Introduction to Bayesian Networks*, Springer-Verlag, NY.
4. Jensen, F. V., K. G. Olesen and S. K. Andersen (1990a), "An algebra of Bayesian belief universes for knowledge-based systems," *Networks*, **20**(5), 637–659.
5. Jensen, F. V., S. L. Lauritzen and K. G. Olesen (1990b), "Bayesian updating in causal probabilistic networks by local computation," *Computational Statistics Quarterly*, **4**, 269–282.

6. Lauritzen, S. L. and D. J. Spiegelhalter (1988), "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *Journal of Royal Statistical Society, Series B*, **50**(2), 157-224.

7. Lauritzen, S. L. and F. V. Jensen (1996), "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, Aalborg, Denmark.

8. Lauritzen, S. L. and P. P. Shenoy (1996) "Computing marginals using local computation", Working Paper No 267, School of Business, University of Kansas, Lawrence, KS. Available by anonymous ftp from ftp.bschool.ukans.edu/data/pub/pshenoy/wp267.ps.

9. Pearl, J. (1986), "Fusion, propagation and structuring in belief networks," *Artificial Intelligence*, **29**, 241–288.

10. Shafer, G. (1996), *Probabilistic Expert Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

11. Shenoy, P. P. (1992), "Valuation-based systems: A framework for managing uncertainty in expert systems," in L. A. Zadeh and J. Kacprzyk (eds.), *Fuzzy Logic for the Management of Uncertainty*, 83–104, John Wiley & Sons, New York, NY.

12. Shenoy, P. P. (1997), "Binary join trees for computing marginals in the Shenoy-Shafer architecture," *International Journal of Approximate Reasoning*, in press.

13. Shenoy, P. P. and G. Shafer (1986), "Propagating belief functions using local computation," *IEEE Expert*, **1**(3), 43-52.

14. Shenoy, P. P. and G. Shafer (1990), "Axioms for probability and belief-function propagation," in R. D. Shachter, T. S. Levitt, J. F. Lemmer and L. N. Kanal (eds.), *Uncertainty in Artificial Intelligence, 4*, 169-198, North-Holland, Amsterdam. Reprinted in Shafer, G. and J. Pearl, eds. (1990), *Readings in Uncertain Reasoning*, 575–610, Morgan Kaufmann, San Mateo, CA.

15. Tarjan, R. E. and M. Yannakakis (1984), "Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM Journal of Computing*, **13**, 566-579.

APPENDIX. COUNTING STORAGE AND OPERATIONS

Figure A.1. A Junction Tree for the Stud Farm Problem

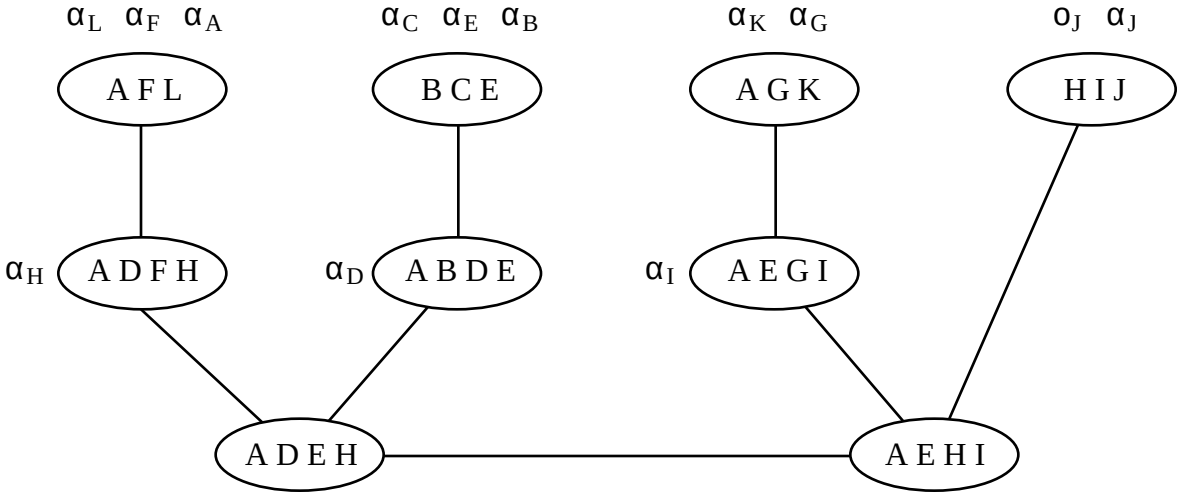


Figure A.2. A Binary Join tree for the Stud Farm Problem

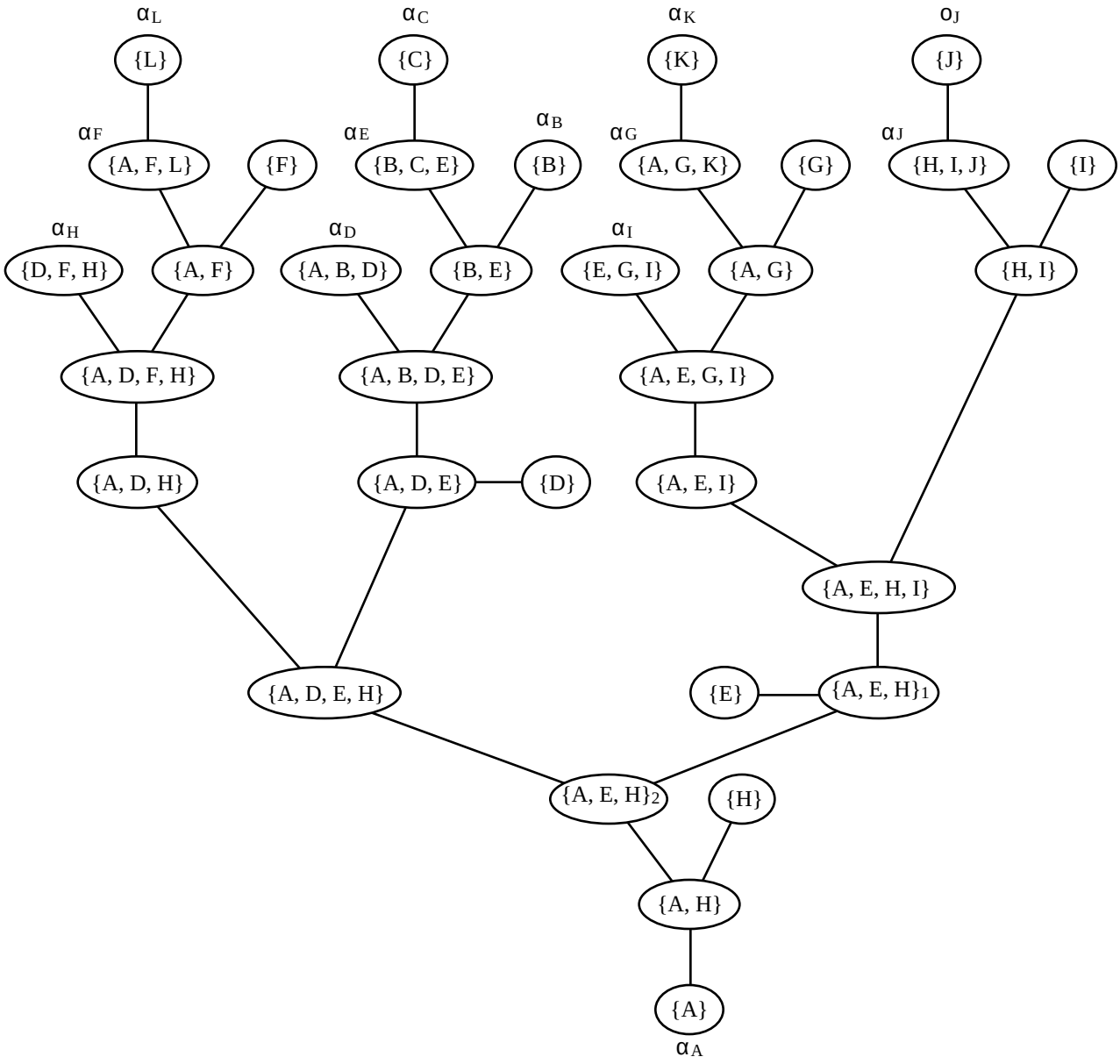


Figure A.3. A Junction Tree for the Genetic Reproduction Problem

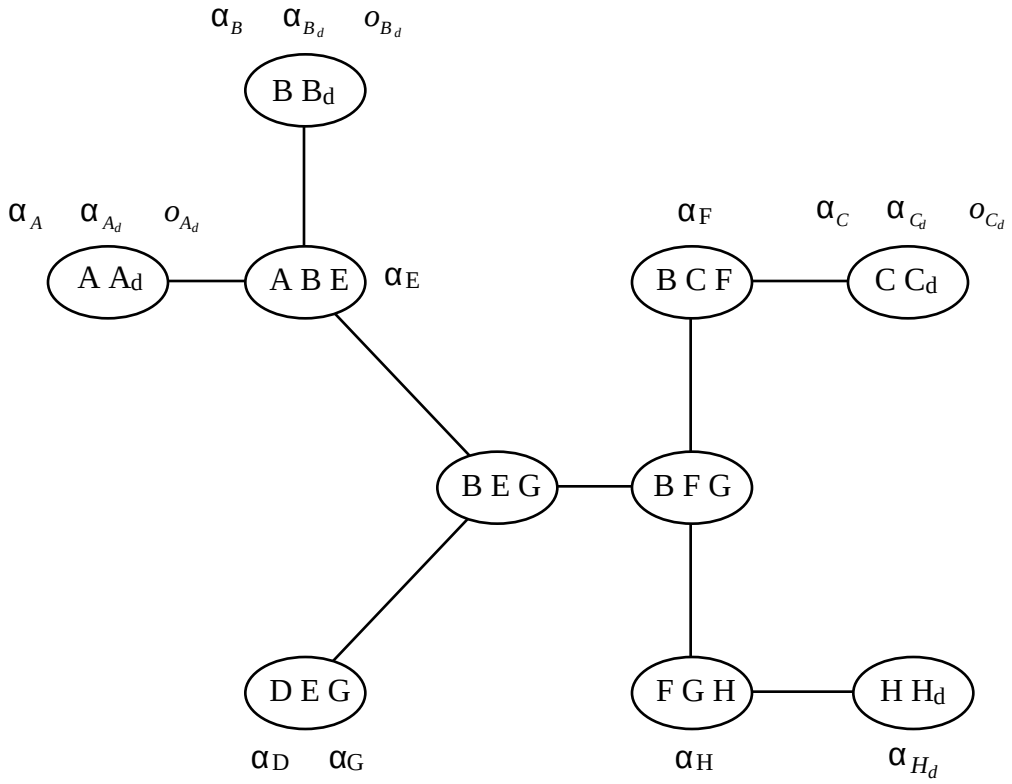


Figure A.4. A Binary Join Tree for the Genetic Reproduction Problem

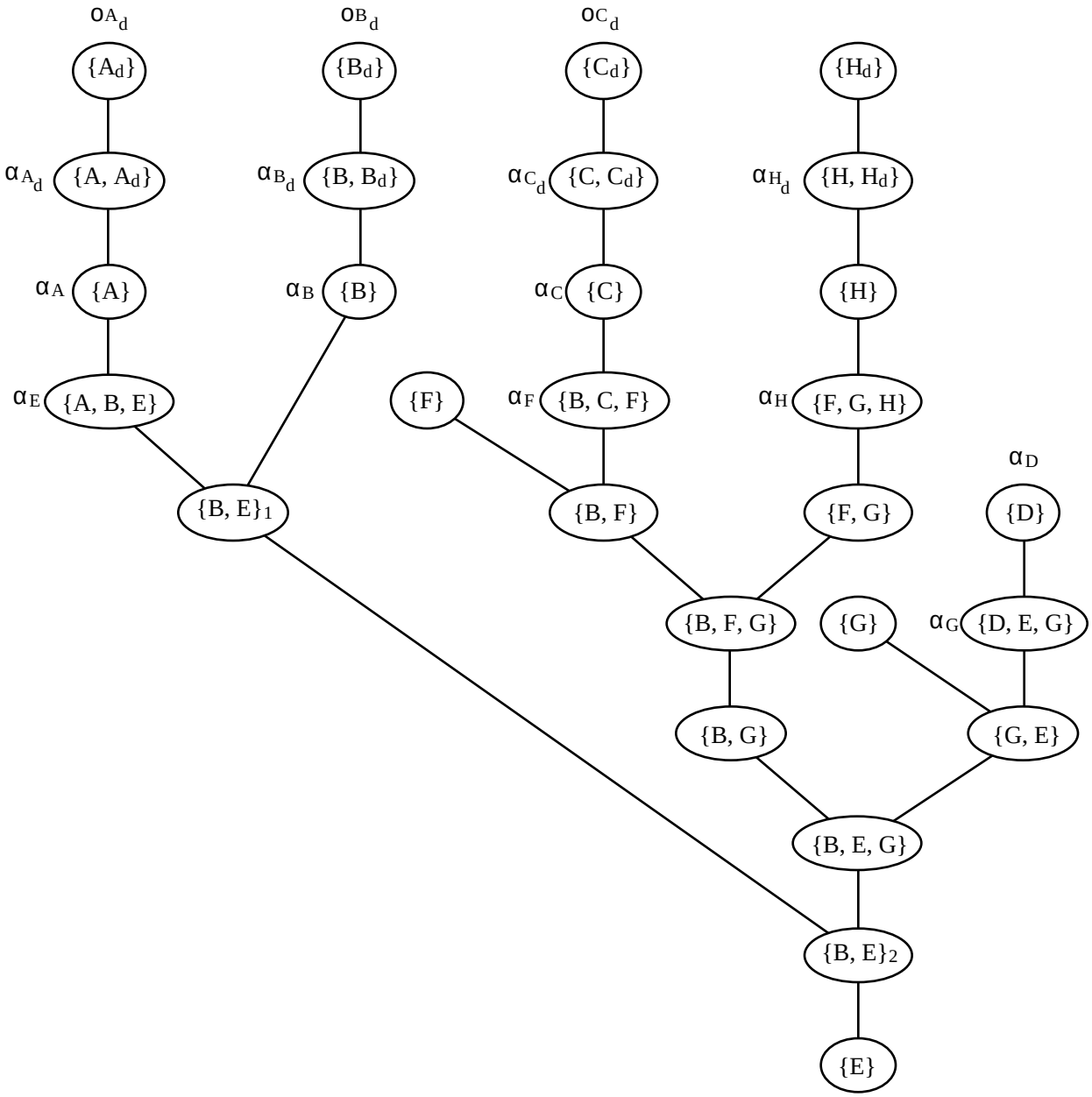


Table A.1. Storage Requirements for Chest Clinic Problem with Evidence for A and D

Storage Requirements # fpn	LS	Hugin	Storage Requirements # fpn	SS
Input Storage	36	36	At Nodes with 1 Storage Register	42
Evidence Storage	4	4	At Nodes with 2 Storage Registers	14
Output Storage	16	16		
Clique Storage	40	40	At Separators with 1 Storage Register	10
Separator Storage		16	At Separators with 2 Storage Registers	92
TOTAL Storage	96	112	TOTAL Storage	158

LS and Hugin:

Input storage (α, σ, τ, λ, $\beta\xi$, ε, δ): 2 + 2 + 4 + 4 + 4 + 4 + 8 + 8 = 36

Evidence Storage (o_A, o_D): 4

Output Storage (8 variables with 2 each): 16

6 Cliques (2 with 2 variables, 4 with 3 variables): (2*4) + (4*8) = 40

5 Separators (2 with 1 variable, 3 with 2 variables): (2*2) + (3*4) = 16

SS:

3 Nodes with ≥ 2 storage registers: (4 input potentials at A (α, o_A), S (σ), D (o_D) and 3 output potentials): 14

11 Nodes with 1 storage register: (only output register T, E, X, L, B): 5*2 = 10

(only input register AT, SL, EX, SB, TLE, EBD): (4*4) + (2*8) = 32

6 Nodes with 0 storage registers: 0

4 Separators with 1 register (SLB—SB = SB, LB—B = B, L—LE = L, X—EX = X): 10

15 separators with 2 registers (7 with 1 variable, 8 with 2 variables): 2*((7*2) + (8*4)) = 92

Table A.2. Storage Requirements for Stud Farm Problem

Storage Requirements # fpn	LS	Hugin	Storage Requirements # fpn	SS
Input Storage	70	70	At Nodes with 1 Storage Register	72
Evidence Storage	3	3	At Nodes with 2 Storage Registers	26
Output Storage	25	25		
Clique Storage	116	116	At Separators with 1 Storage Register	36
Separator Storage		48	At Separators with 2 Storage Registers	242
TOTAL Storage	214	262	TOTAL Storage	376

LS and Hugin:

Input storage (12 potentials, 5 with 2 fpn, 6 with 8 fpn, 1 with 12 fpn): (5*2) + (6*8) + (1*12) = 70

Evidence Storage (1 potential o_J): 3

Output Storage (11 variables with 2 fpn, 1 with 3 fpn): 22 + 3 = 25

9 Clique storage (3 with 8 states, 1 with 12 states, 5 with 16 states) (3 * 8) + (1*12) + (5 *16) = 24 + 12 + 80 = 116

8 Separator storage (4 with 4 states, 4 with 8 states): (4 * 4) + (4 * 8) = 16 + 32 = 48

SS:

13 Nodes with 1 storage register (only an output register): F, G, I, D, E, H. Storage = 2*6 = 12

(only input register): AFL, BCE, AGK, HIJ, DFH, ABD, EGI

storage = (6*8) + (1*12) = 48 + 12 = 60

6 Nodes with 2 storage registers (L, C, K, B, A, J): Storage = 2*2*5 + 2*3*1 = 26

15 Nodes with 0 storage registers: 0

9 Separators with 1 register (F—AF = F, G—AG = G, I—HI = I, D—ADE = D, H—AH = H, E—AEH_1 = E, DFH—ADFH = DFH, ABD—ABDE= ABD, EGI—AEGI = EGI): Storage = (2*6) + (3*8) = 12 + 24 = 36

24 Separators with 2 registers (6 with 1 variable, 9 with 2 variables, and 9 with 3 variables): Storage = 2*(13 + 36 + 72) = 242

Table A.3. Storage Requirements for Genetic Reproduction Problem

Storage Requirements # fpn	LS	Hugin	Storage Requirements # fpn	SS
Input Storage	144	144	At Nodes with 1 Storage Register	146
Evidence Storage	6	6	At Nodes with 2 Storage Registers	36
Output Storage	32	32		
Clique Storage	186	186	At Separators with 1 Storage Register	11
Separator Storage		57	At Separators with 2 Storage Registers	264
TOTAL Storage	368	425	TOTAL Storage	457

LS and Hugin:

Input storage (4 potentials with 3 fpn, 4 potentials with 6 fpn, 4 with 27 fpn): (4*3) + (4*6) + (4*27) = 12 + 24 + 108 = 144

Evidence Storage (3 potentials with 2 fpn): 6

Output Storage (4 variables with 2 fpn, 8 variables with 3 fpn): 8 + 24 = 32

10 Clique storage (4 with 6 states, 6 with 27 states) (4 * 6) + (6 *27) = 24 + 162 = 186

9 Separator storage (4 with 3 states, 5 with 9 states): (4 * 3) + (5 * 9) = 12 + 45 = 57

SS:

13 Nodes with 1 storage register (1 node with 2 fpn, 4 nodes with 3 fpn, 4 nodes with 6 fpn, 4 with 27 fpn): (1*2) + (4*3) + (4*6) + (4*27) = 2 + 12 + 24 + 108 = 146

7 Nodes with 2 storage registers (3 with 2 fpn, 4 with 3 fpn): 2*((3*2) + (4*3))= 36

8 Nodes with 0 storage registers: 0

4 Separators with 1 register (1 with 2 fpn, 3 with 3 fpn): (1*2) + (3*3) = 2 + 9 = 11

23 Separators with 2 registers (3 with 2 fpn, 9 with 3 fpn, and 11 with 9 fpn): 2*((3*2) + (9*3) + (11*9)) = 2*(6 + 27 + 99) = 2*132 = 264

Table A.4. Computational Details in the LS Architecture for the Chest Clinic Problem Using the Junction Tree in Figure 5 with No Evidence

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{A, T}	$\chi_1 = \alpha \quad \tau$		4	
{S, L, B}	$\chi_4 = \sigma \quad \lambda \quad \beta$		16	
Inward propagation (root node is {A, T}):				
{E, X}	$\chi_6' = \chi_6 / \chi_6^{\downarrow \{E\}}$	2		4
{E, D, B}	$\chi_5' = \chi_5 / \chi_5^{\downarrow \{E, B\}}$	4		8
{S, L, B}	$\chi_4' = \chi_4 / \chi_4^{\downarrow \{L, B\}}$	4		8
{L, E, B}	$\chi_3' = [\chi_4^{\downarrow \{L, B\}} \quad \chi_5^{\downarrow \{E, B\}}] / [\chi_4^{\downarrow \{L, B\}} \quad \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}}$	4	8	8
{T, L, E}	$\chi_2' = [\chi_2 \quad [\chi_4^{\downarrow \{L, B\}} \quad \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L,E\}} \quad \chi_6^{\downarrow \{E\}}] / [\chi_2 \quad [\chi_4^{\downarrow \{L, B\}} \quad \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}} \quad \chi_6^{\downarrow \{E\}}]^{\downarrow \{T\}}$	6	16	8
{A, T}	$\chi_1' = \chi_1 \quad [\chi_2 \quad [[\chi_4^{\downarrow \{L, B\}} \quad \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}} \quad \chi_6^{\downarrow \{E\}}]]^{\downarrow \{T\}} = \chi_1''$		4	
Outward propagation:				
{A, T}	$\chi_1''^{\downarrow \{T\}}$	2		
{T, L, E}	$\chi_2'' = \chi_2' \quad \chi_1''^{\downarrow \{T\}}, \chi_2''^{\downarrow \{L, E\}}, \chi_2''^{\downarrow \{E\}}$	10	8	
{L, E, B}	$\chi_3'' = \chi_3' \quad \chi_2''^{\downarrow \{L, E\}}, \chi_3''^{\downarrow \{L, B\}}, \chi_3''^{\downarrow \{E, B\}}$	8	8	
{S, L, B}	$\chi_4'' = \chi_4' \quad \chi_3''^{\downarrow \{L, B\}}$		8	
{E, D, B}	$\chi_5'' = \chi_5' \quad \chi_3''^{\downarrow \{E, B\}}$		8	
{E, X}	$\chi_6'' = \chi_6' \quad \chi_2''^{\downarrow \{E\}}$		4	
Computing marginals of singletons:				
{A, T}	$\chi_1''^{\downarrow \{A\}}$ (marginal for A)	2		
{S, L, B}	$\chi_4''^{\downarrow \{S\}}$ (marginal for S)	6		
{A, T}	$\chi_1''^{\downarrow \{T\}}$ (marginal for T)	2		
{T, L, E}	$\chi_2''^{\downarrow \{L\}}$ (marginal for L)	6		
{L, E, B}	$\chi_3''^{\downarrow \{B\}}$ (marginal for B)	6		
{E, X}	$\chi_6''^{\downarrow \{E\}}$ (marginal for E)	2		
{E, B, D}	$\chi_5''^{\downarrow \{D\}}$ (marginal for D)	6		
{E, X}	$\chi_6''^{\downarrow \{X\}}$ (marginal for X)	2		
	TOTALS	72	84	36

Table A.5. Computational Details in the LS Architecture for the Chest Clinic Problem Using the Junction Tree in Figure 5 with Evidence for A = a, D = d

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{A, T}	$\chi_1 = \alpha \ o_A \ \tau$		8	
{S, L, B}	$\chi_4 = \sigma \ \lambda \ \beta$		16	
{E, D, B}	$\chi_5 = \delta \ o_D$		8	
Inward propagation (root node is {A, T}):				
{E, X}	$\chi_6' = \chi_6 / \chi_6^{\downarrow \{E\}}$	2		4
{E, D, B}	$\chi_5' = \chi_5 / \chi_5^{\downarrow \{E, B\}}$	4		8
{S, L, B}	$\chi_4' = \chi_4 / \chi_4^{\downarrow \{L, B\}}$	4		8
{L, E, B}	$\chi_3' = [\chi_4^{\downarrow \{L, B\}} \ \chi_5^{\downarrow \{E, B\}}] / [\chi_4^{\downarrow \{L, B\}} \ \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}}$	4	8	8
{T, L, E}	$\chi_2' = [\chi_2 \ [\chi_4^{\downarrow \{L, B\}} \ \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L,E\}} \ \chi_6^{\downarrow \{E\}}] / [\chi_2 \ [\chi_4^{\downarrow \{L, B\}} \ \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}} \ \chi_6^{\downarrow \{E\}}]^{\downarrow \{T\}}$	6	16	8
{A, T}	$\chi_1' = \chi_1 \ [\chi_2 \ [[\chi_4^{\downarrow \{L, B\}} \ \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}} \ \chi_6^{\downarrow \{E\}}]]^{\downarrow \{T\}} = \chi_1''$		4	
Outward propagation:				
{A, T}	$\chi_1''^{\downarrow \{T\}}$	2		
{T, L, E}	$\chi_2'' = \chi_2' \ \chi_1''^{\downarrow \{T\}}, \chi_2''^{\downarrow \{L, E\}}, \chi_2''^{\downarrow \{E\}}$	10	8	
{L, E, B}	$\chi_3'' = \chi_3' \ \chi_2''^{\downarrow \{L, E\}}, \chi_3''^{\downarrow \{L, B\}}, \chi_3''^{\downarrow \{E, B\}}$	8	8	
{S, L, B}	$\chi_4'' = \chi_4' \ \chi_3''^{\downarrow \{L, B\}}$		8	
{E, D, B}	$\chi_5'' = \chi_5' \ \chi_3''^{\downarrow \{E, B\}}$		8	
{E, X}	$\chi_6'' = \chi_6' \ \chi_2''^{\downarrow \{E\}}$		4	
Computing marginals of singletons:				
{A, T}	$\chi_1''^{\downarrow \{A\}}$ (marginal for A)	2		
{S, L, B}	$\chi_4''^{\downarrow \{S\}}$ (marginal for S)	6		
{A, T}	$\chi_1''^{\downarrow \{T\}}$ (marginal for T)	2		
{T, L, E}	$\chi_2''^{\downarrow \{L\}}$ (marginal for L)	6		
{L, E, B}	$\chi_3''^{\downarrow \{B\}}$ (marginal for B)	6		
{E, X}	$\chi_6''^{\downarrow \{E\}}$ (marginal for E)	2		
{E, B, D}	$\chi_5''^{\downarrow \{D\}}$ (marginal for D)	6		
{E, X}	$\chi_6''^{\downarrow \{X\}}$ (marginal for X)	2		
TOTALS		72	96	36

Table A.6. Computational Details in the LS Architecture for the Chest Clinic Problem Using the Junction Tree in Figure 5 with the Evidence for A, D, S, and X

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{A, T}	$\chi_1 = \alpha \circ_A \tau$		8	
{S, L, B}	$\chi_4 = \sigma \lambda \beta \circ_S$		24	
{E, D, B}	$\chi_5 = \delta \circ_D$		8	
{E, X}	$\chi_6 = \xi \circ_X$		4	
Inward propagation (root node is {A, T}):				
{E, X}	$\chi_6' = \chi_6 / \chi_6^{\downarrow \{E\}}$	2		4
{E, D, B}	$\chi_5' = \chi_5 / \chi_5^{\downarrow \{E, B\}}$	4		8
{S, L, B}	$\chi_4' = \chi_4 / \chi_4^{\downarrow \{L, B\}}$	4		8
{L, E, B}	$\chi_3' = [\chi_4^{\downarrow \{L, B\}} \chi_5^{\downarrow \{E, B\}}] / [\chi_4^{\downarrow \{L, B\}} \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}}$	4	8	8
{T, L, E}	$\chi_2' = [\chi_2 [\chi_4^{\downarrow \{L, B\}} \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L,E\}} \chi_6^{\downarrow \{E\}}] / [\chi_2 [\chi_4^{\downarrow \{L, B\}} \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}} \chi_6^{\downarrow \{E\}}]^{\downarrow \{T\}}$	6	16	8
{A, T}	$\chi_1' = \chi_1 [\chi_2 [[\chi_4^{\downarrow \{L, B\}} \chi_5^{\downarrow \{E, B\}}]^{\downarrow \{L, E\}} \chi_6^{\downarrow \{E\}}]]^{\downarrow \{T\}} = \chi_1''$		4	
Outward propagation:				
{A, T}	$\chi_1''^{\downarrow \{T\}}$	2		
{T, L, E}	$\chi_2'' = \chi_2' \chi_1''^{\downarrow \{T\}}, \chi_2''^{\downarrow \{L, E\}}, \chi_2''^{\downarrow \{E\}}$	10	8	
{L, E, B}	$\chi_3'' = \chi_3' \chi_2''^{\downarrow \{L, E\}}, \chi_3''^{\downarrow \{L, B\}}, \chi_3''^{\downarrow \{E, B\}}$	8	8	
{S, L, B}	$\chi_4'' = \chi_4' \chi_3''^{\downarrow \{L, B\}}$		8	
{E, D, B}	$\chi_5'' = \chi_5' \chi_3''^{\downarrow \{E, B\}}$		8	
{E, X}	$\chi_6'' = \chi_6' \chi_2''^{\downarrow \{E\}}$		4	
Computing marginals of singletons:				
{A, T}	$\chi_1''^{\downarrow \{A\}}$ (marginal for A)	2		
{S, L, B}	$\chi_4''^{\downarrow \{S\}}$ (marginal for S)	6		
{A, T}	$\chi_1''^{\downarrow \{T\}}$ (marginal for T)	2		
{T, L, E}	$\chi_2''^{\downarrow \{L\}}$ (marginal for L)	6		
{L, E, B}	$\chi_3''^{\downarrow \{B\}}$ (marginal for B)	6		
{E, X}	$\chi_6''^{\downarrow \{E\}}$ (marginal for E)	2		
{E, B, D}	$\chi_5''^{\downarrow \{D\}}$ (marginal for D)	6		
{E, X}	$\chi_6''^{\downarrow \{X\}}$ (marginal for X)	2		
TOTALS		72	108	36

Table A.7. Computational Details in the Hugin Architecture for the Chest Clinic Problem in the Junction Tree in Figure 5 with Evidence for A and D

Computation		# binary arithmetic operations		
At Node/Edge	Details	+	×	÷
At the beginning:				
{A, T}	$\chi_1 = \alpha \quad o_A \quad \tau$		8	
{S, L, B}	$\chi_4 = \sigma \quad \lambda \quad \beta$		16	
{E, D, B}	$\chi_5 = \delta \quad o_D$		8	
Inward propagation (root node is {A, T}):				
{E, X}	$\chi_6^{\downarrow \{E\}}$	2		
{E, B, D}	$\chi_5^{\downarrow \{E, B\}}$	4		
{S, L, B}	$\chi_4^{\downarrow \{L, B\}}$	4		
{L, E, B}	$\chi_3' = [\chi_4^{\downarrow \{L, B\}} \quad \chi_5^{\downarrow \{E, B\}}], \chi_3'^{\downarrow \{L, E\}}$	4	8	
{T, L, E}	$\chi_2' = \chi_2 \quad \chi_3'^{\downarrow \{L, E\}} \quad \chi_6^{\downarrow \{E\}}, \chi_2'^{\downarrow \{T\}}$	6	16	
{A, T}	$\chi_1' = \chi_1 \quad \chi_2'^{\downarrow \{T\}} = \chi_1''$		4	
Outward propagation:				
{A, T}	$\chi_1''^{\downarrow \{T\}}$ (marginal for T)	2		
AT–TLE	$\chi_1''^{\downarrow \{T\}} / \chi_2'^{\downarrow \{T\}}$			2
{T, L, E}	$\chi_2'' = \chi_2' \quad [\chi_1''^{\downarrow \{T\}} / \chi_2'^{\downarrow \{T\}}], \chi_2''^{\downarrow \{E\}}$ (marginal for E), $\chi_2''^{\downarrow \{L, E\}}$	10	8	
TLE–LEB	$\chi_2''^{\downarrow \{L, E\}} / \chi_3'^{\downarrow \{L, E\}}$			4
{L, E, B}	$\chi_3'' = \chi_3' \quad [\chi_2''^{\downarrow \{L, E\}} / \chi_3'^{\downarrow \{L, E\}}], \chi_3''^{\downarrow \{L, B\}}, \chi_3''^{\downarrow \{E, B\}}$	8	8	
LEB–SLB	$\chi_3''^{\downarrow \{L, B\}} / \chi_4^{\downarrow \{L, B\}}$			4
{S, L, B}	$\chi_4'' = \chi_4' \quad [\chi_3''^{\downarrow \{L, B\}} / \chi_4^{\downarrow \{L, B\}}]$		8	
LEB–EBD	$\chi_3''^{\downarrow \{E, B\}} / \chi_5^{\downarrow \{E, B\}}$			4
{E, D, B}	$\chi_5'' = \chi_5' \quad [\chi_3''^{\downarrow \{E, B\}} / \chi_5^{\downarrow \{E, B\}}]$		8	
TLE–EX	$\chi_2''^{\downarrow \{E\}} / \chi_6^{\downarrow \{E\}}$			2
{E, X}	$\chi_6'' = \chi_6' \quad [\chi_2''^{\downarrow \{E\}} / \chi_6^{\downarrow \{E\}}]$		4	
Computing marginals of singletons:				
{A, T}	$\chi_1''^{\downarrow \{A\}}$ (marginal for A)	2		
TLE–LEB	$[\chi_2''^{\downarrow \{L, E\}}]^{\downarrow \{L\}}$ (marginal for L)	2		
LEB–SLB	$[\chi_3''^{\downarrow \{L, B\}}]^{\downarrow \{B\}}$ (marginal for B)	2		
{S, L, B}	$\chi_4''^{\downarrow \{S\}}$ (marginal for S)	6		
{E, B, D}	$\chi_5''^{\downarrow \{D\}}$ (marginal for D)	6		
{E, X}	$\chi_6''^{\downarrow \{X\}}$ (marginal for X)	2		
TOTALS		60	96	16

Table A.8. Computational Details in the SS Architecture for the Chest Clinic Problem using the Binary Join Tree in Figure 16 with Evidence for A and D

Computation		# binary arithmetic operations		
At Node	Details	+	×	÷
Propagation of Messages				
{A}	$\alpha \quad o_A = \mu^{A \to AT}$		2	
{A, T}	$\mu^{AT \to T} = (\mu^{A \to AT} \quad \tau)^{\downarrow \{T\}} = \mu^{T \to TE}$	2	4	
{S, L}	$\mu^{SL \to SLB} = \sigma \quad \lambda$		4	
{S, L, B}	$\mu^{SLB \to LB} = (\mu^{SL \to SLB} \quad \beta)^{\downarrow \{L, B\}} = \mu^{LB \to LEB}$	4	8	
{E, B D}	$\mu^{EBD \to EB} = (o_D \quad \delta)^{\downarrow \{E, B\}} = \mu^{EB \to LEB}$	4	8	
{E, X}	$\mu^{EX \to E} = \zeta^{\downarrow E} = \mu^{E \to TE}$	2		
{L, E, B}	$\mu^{LEB \to LE} = (\mu^{EB \to LEB} \quad \mu^{LB \to LEB})^{\downarrow \{L, E\}} = \mu^{LE \to TLE}$	4	8	
{T, L, E}	$\mu^{TLE \to TE} = (\varepsilon \quad \mu^{LE \to TLE})^{\downarrow \{T, E\}}$	4	8	
{T, E}	$\mu^{TE \to E} = (\mu^{T \to TE} \quad \mu^{TLE \to TE})^{\downarrow E} = \mu^{E \to EX}$,	2	4	
	$\mu^{TE \to T} = (\mu^{E \to TE} \quad \mu^{TLE \to TE})^{\downarrow T} = \mu^{T \to AT}$,	2	4	
	$\mu^{TE \to TLE} = \mu^{T \to TE} \quad \mu^{E \to TE}$		4	
{A, T}	$\mu^{AT \to A} = (\mu^{T \to AT} \quad \tau)^{\downarrow A}$	2	4	
{T, L, E}	$\mu^{TLE \to LE} = (\mu^{TE \to TLE} \quad \varepsilon)^{\downarrow \{L, E\}} = \mu^{LE \to LEB}$	4	8	
{L, E}	$\mu^{LE \to L} = (\mu^{TLE \to LE} \quad \mu^{LEB \to LE})^{\downarrow L}$ (marg. for L)	2	4	
{E, X}	$\mu^{EX \to X} = (\mu^{TE \to E} \quad \xi)^{\downarrow X}$ (marg. for X)	2	4	
{L, E, B}	$\mu^{LEB \to EB} = (\mu^{LE \to LEB} \quad \mu^{LB \to LEB}]^{\downarrow \{E, B\}}$,	8	16	
	$\mu^{LEB \to LB} = (\mu^{LE \to LEB} \quad \mu^{EB \to LEB})^{\downarrow \{L, B\}}$			
{L, B}	$\mu^{LB \to B} = (\mu^{SLB \to LB} \quad \mu^{LEB \to LB})^{\downarrow B}$ (marg. for B)	2	4	
{S, L, B}	$\mu^{SLB \to SL} = (\mu^{LB \to SLB} \quad \beta)^{\downarrow \{S, L\}}$	4	8	
{S, L}	$\mu^{SL \to S} = (\mu^{SLB \to SL} \quad \lambda)^{\downarrow S}$	2	4	
{E, B, D}	$\mu^{EBD \to D} = (\mu^{EB \to EBD} \quad \delta)^{\downarrow D}$	6	8	
Computation of Marginals:				
{A}	$\mu^{AT \to A} \quad (\alpha \quad o_A)$		2	
{S}	$\sigma \quad \mu^{SL \to S}$		2	
{T}	$\mu^{AT \to T} \quad \mu^{TE \to T}$		2	
{E}	$\mu^{EX \to E} \quad \mu^{TE \to E}$		2	
{D}	$\mu^{EBD \to D} \quad o_D$		2	
	TOTALS	56	124	

Table A.9. Computational Details in the SS Architecture for the Chest Clinic Problem Using the Junction Tree in Figure 5 with Evidence for A and D

Computation		# binary arithmetic operations		
At Node/Edge	Details	+	×	÷
Propagation of Messages:				
{A, T}	$\mu^{AT\to TLE} = (\alpha \ \tau \ o_A)^{\downarrow T}$	2	8	
{S, L, B}	$\mu^{SLB\to LEB} = (\sigma \ \lambda \ \beta)^{\downarrow \{L, B\}}$	4	16	
{E, D, B}	$\mu^{EBD\to LEB} = (\delta \ o_D)^{\downarrow \{E, B\}}$	4	8	
{E, X}	$\mu^{EX\to TLE} = \zeta^{\downarrow E}$	2		
{L, E, B}	$\mu^{LEB\to TLE} = (\mu^{EBD\to LEB} \ \mu^{SLB\to LEB})^{\downarrow \{L, E\}}$	4	8	
{T, L, E}	$\mu^{TLE\to LEB} = (\varepsilon \ \mu^{AT\to TLE} \ \mu^{LEB\to TLE})^{\downarrow \{L, E\}}$	4	16	
	$\mu^{TLE\to AT} = (\varepsilon \ \mu^{LEB\to TLE} \ \mu^{EX\to TLE})^{\downarrow T}$	6	16	
	$\mu^{TLE\to EX} = (\varepsilon \ \mu^{LEB\to TLE} \ \mu^{AT\to TLE})^{\downarrow \{E\}}$	6	16	
{L, E, B}	$\mu^{LEB\to EBD} = (\mu^{TLE\to LEB} \ \mu^{SLB\to LEB})^{\downarrow \{E, B\}}$	4	8	
	$\mu^{LEB\to LB} = (\mu^{LE\to LEB} \ \mu^{EB\to LEB})^{\downarrow \{L, B\}}$	4	8	
Computation of Marginals:				
{A, T}	$(\mu^{TLE\to AT} \ \chi_1)^{\downarrow A}$ (marg. for A)	2	4	
{S, L, B}	$(\mu^{LEB\to SLB} \ \chi_4)^{\downarrow S}$ (marg. for S)	6	8	
AT—TLE	$\mu^{TLE\to AT} \ \mu^{AT\to TLE}$ (marg. for T)		2	
TLE—LEB	$(\mu^{TLE\to LEB} \ \mu^{LEB\to TLE})^{\downarrow L}$ (marg. for L)	2	4	
LEB—SLB	$(\mu^{SLB\to LEB} \ \mu^{LEB\to SLB})^{\downarrow B}$ (marg. for B)	2	4	
TLE—EX	$\mu^{TLE\to E} \ \mu^{EX\to E}$ (marg. for E)		2	
{E, D, B}	$(\mu^{EB\to EBD} \ \chi_6)^{\downarrow D}$ (marg. for D)	6	8	
{E, X}	$(\mu^{TLE\to EX} \ \xi)^{\downarrow X}$ (marg. for X)	2	4	
TOTALS		60	140	

Table A.10. Computational Details in the Hugin Architecture for the Chest Clinic Problem using the Binary Join Tree in Figure 16 with Evidence for A and D

Computation		# binary arithmetic operations		
At Node/Edge	Details	+	×	÷
At the beginning:				
{A}	$\chi_1 = \alpha \quad o_A$		2	
Inward propagation (with root node {A}):				
{E, X}	$\chi_2 = \xi = \chi_2', \chi_2'^{\downarrow \{E\}}$	2		
{E, D, B}	$\chi_3 = \delta, \chi_3' = \chi_3 \quad o_D, \chi_3'^{\downarrow \{E, B\}}$	4	8	
{S, L}	$\chi_4 = \lambda, \chi_4' = \chi_4 \quad \sigma$		4	
{S, L, B}	$\chi_5' = \chi_4' \quad \beta, \chi_5'^{\downarrow \{L, B\}}$	4	8	
{L, E, B}	$\chi_6' = \chi_3'^{\downarrow \{E, B\}} \quad \chi_5'^{\downarrow \{L, B\}}, \chi_6'^{\downarrow \{L, E\}}$	4	8	
{T, L, E}	$\chi_7 = \varepsilon, , \chi_7' = \chi_7 \quad \chi_6'^{\downarrow \{L, E\}}, \chi_7'^{\downarrow \{T, E\}}$	4	8	
{T, E}	$\chi_8' = \chi_2'^{\downarrow \{E\}} \quad \chi_7'^{\downarrow \{T, E\}}, \chi_8'^{\downarrow \{T\}}$	2	4	
{A, T}	$\chi_9 = \tau, \chi_9' = \chi_9 \quad \chi_8'^{\downarrow \{T\}}, \chi_9'^{\downarrow \{A\}}$	2	4	
{A}	$\chi_1' = \chi_1 \quad \chi_9'^{\downarrow \{A\}} = \chi_1''$ (marginal for A)		2	
Outward propagation:				
A—AT	$\chi_1'' / \chi_9'^{\downarrow \{A\}}$			2
{A, T}	$\chi_9'' = \chi_9' \quad [\chi_1'' / \chi_9'^{\downarrow \{A\}}], \chi_9''^{\downarrow \{T\}}$	2	4	
AT—T	$\chi_9''^{\downarrow \{T\}} / \chi_8'^{\downarrow \{T\}}$			2
{T}	$\chi_{13}'' = \chi_8'^{\downarrow \{T\}} \quad [\chi_9''^{\downarrow \{T\}} / \chi_8'^{\downarrow \{T\}}]$ (marginal for T)		2	
T—TE	$\chi_{13}'' / \chi_8'^{\downarrow \{T\}}$			2
{T, E}	$\chi_8'' = \chi_8' \quad [\chi_{13}'' / \chi_8'^{\downarrow \{T\}}], \chi_8''^{\downarrow \{E\}}$	2	4	
TE—E	$\chi_8''^{\downarrow \{E\}} / \chi_2'^{\downarrow \{E\}}$			2
{E}	$\chi_{14}'' = \chi_2'^{\downarrow \{E\}} \quad [\chi_8''^{\downarrow \{E\}} / \chi_2'^{\downarrow \{E\}}]$ (marginal for E)		2	
E—EX	$\chi_{14}'' / \chi_2'^{\downarrow \{E\}}$			2
{E, X}	$\chi_2'' = \chi_2' \quad [\chi_{14}'' / \chi_2'^{\downarrow \{E\}}], \chi_2''^{\downarrow \{X\}}$ (marginal for X)	2	4	
TE—TLE	$\chi_8'' / \chi_7'^{\downarrow \{T, E\}}$			4
{T, L, E}	$\chi_7'' = \chi_7' \quad [\chi_8'' / \chi_7'^{\downarrow \{T, E\}}], \chi_7''^{\downarrow \{L, E\}}$	4	8	
TLE—LE	$\chi_7''^{\downarrow \{L, E\}} / \chi_6'^{\downarrow \{L, E\}}$			4
{L, E}	$\chi_{15}'' = \chi_6'^{\downarrow \{L, E\}} \quad [\chi_7''^{\downarrow \{L, E\}} / \chi_6'^{\downarrow \{L, E\}}], \chi_{15}''^{\downarrow \{L\}}$ (marginal for L)	2	4	
LE—LEB	$\chi_{15}'' / \chi_6'^{\downarrow \{L, E\}}$			4

{L, E, B}	$\chi_6'' = \chi_6' \quad [\chi_{15}'' / \chi_6'^{\downarrow \{L, E\}}], \chi_6''^{\downarrow \{L, B\}}, \chi_6''^{\downarrow \{E, B\}}$	8	8	
LEB—LB	$\chi_6''^{\downarrow \{L, B\}} / \chi_5'^{\downarrow \{L, B\}}$			4
{L, B}	$\chi_{16}'' = \chi_5'^{\downarrow \{L, B\}} \quad [\chi_6''^{\downarrow \{L, B\}} / \chi_5'^{\downarrow \{B\}}], \chi_{16}''^{\downarrow \{B\}}$ (marginal for B)	2	4	
LB—SLB	$\chi_{16}'' / \chi_5'^{\downarrow \{L, B\}}$			4
{S, L, B}	$\chi_5'' = \chi_5' \quad [\chi_{16}'' / \chi_5'^{\downarrow \{L, B\}}], \chi_5''^{\downarrow \{S, L\}}, \chi_5''^{\downarrow \{S, B\}}$	8	8	
SLB—SL	$\chi_5''^{\downarrow \{S, L\}} / \chi_4'$			4
{S, L}	$\chi_4'' = \chi_4' \quad [\chi_5''^{\downarrow \{S, L\}} / \chi_4'], \chi_4''^{\downarrow \{S\}}$	2	4	
SL—S	$\chi_4''^{\downarrow \{S\}} / \sigma$			2
{S}	$\chi_{11}'' = \sigma \quad [\chi_4''^{\downarrow \{S\}} / \sigma]$ (marg. for S)		2	
LEB—EB	$\chi_6''^{\downarrow \{E, B\}} / \chi_3'^{\downarrow \{E, B\}}$			4
{E, B}	$\chi_{10}'' = \chi_3'^{\downarrow \{E, B\}} \quad [\chi_6''^{\downarrow \{E, B\}} / \chi_3'^{\downarrow \{E, B\}}]$		4	
EB—EDB	$\chi_{10}'' / \chi_3'^{\downarrow \{E, B\}}$			4
{E, D, B}	$\chi_3'' = \chi_3' \quad [\chi_{10}'' / \chi_3'^{\downarrow \{E, B\}}], \chi_3''^{\downarrow \{D\}}$	6	8	
EDB—D	$\chi_3''^{\downarrow \{D\}} / o_D$			2
{D}	$\chi_{17}'' = o_D \quad [\chi_3''^{\downarrow \{D\}} / o_D]$ (marg. for D)		2	
	TOTALS	60	116	46

Table A.11. Computational Details in the LS Architecture for the Stud Farm Problem Using the Junction Tree in Figure A.1.

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{A, F, L}	$\chi_1 = \alpha_L \quad \alpha_F \quad \alpha_A$		16	
{B, C, E}	$\chi_5 = \alpha_C \quad \alpha_E \quad \alpha_B$		16	
{A, G, K}	$\chi_8 = \alpha_K \quad \alpha_E$		8	
{H, I, J}	$\chi_9 = o_J \quad \alpha_J$		12	
Inward propagation (root node is {A, F, L}):				
{H, I, J}	$\chi_9' = \chi_9 / \chi_9^{\downarrow \{H, I\}}$	8		12
{A, G, K}	$\chi_8' = \chi_8 / \chi_8^{\downarrow \{A, G\}}$	4		8
{B, C, E}	$\chi_5' = \chi_5 / \chi_5^{\downarrow \{B, E\}}$	4		8
{A, E, G, I}	$\chi_7' = [\alpha_I \quad \chi_8^{\downarrow \{A, G\}}] / [\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}}$	8	16	16
{A, E, H, I}	$\chi_6' = [[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}] / [[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}]^{\downarrow \{A, E,H\}}$	8	16	16
{A, B, D,E}	$\chi_4' = [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}] / [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}]^{\downarrow \{A, D, E\}}$	8	16	16
{A, D,E,H}	$\chi_3' = [[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}]^{\downarrow \{A,E,H\}} [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}]^{\downarrow \{A, D, E\}} / [[[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}]^{\downarrow \{A,E,H\}} \quad [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}]^{\downarrow \{A, D, E\}}]^{\downarrow \{A, D, H\}}$	8	16	16
{A,D,F,H }	$\chi_2' = [\alpha_H \quad [[[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}]^{\downarrow \{A,E,H\}} \quad [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}]^{\downarrow \{A, D, E\}}]^{\downarrow \{A, D, H\}}] / [\alpha_H \quad [[[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}]^{\downarrow \{A,E,H\}} \quad [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}]^{\downarrow \{A, D, E\}}]^{\downarrow \{A, D, H\}}]^{\downarrow \{A, F\}}$	12	16	16
{A, F, L}	$\chi_1' = \chi_1 \quad [\alpha_H \quad [[[\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H,I\}}]^{\downarrow \{A,E,H\}} \quad [\alpha_D \quad \chi_5^{\downarrow \{B, E\}}]^{\downarrow \{A, D, E\}}]^{\downarrow \{A, D, H\}} = \chi_1''$		8	-
Outward propagation:				
{A, F, L}	$\chi_1''^{\downarrow \{A, F\}}$	4		
{A,D,F,H }	$\chi_2'' = \chi_2' \quad \chi_1''^{\downarrow \{A,F\}}, \chi_2''^{\downarrow \{A, D, H\}}$	8	16	

{A, D,E,H}	$\chi_3'' = \chi_3' \ \chi_2''^{\downarrow\{A, D, H\}}, \chi_3''^{\downarrow\{A, D, E\}}, \chi_3''^{\downarrow\{A,E, H\}}$	16	16	
{A, B, D,E}	$\chi_4'' = \chi_4' \ \chi_3''^{\downarrow\{A, D, E\}}, \chi_4''^{\downarrow\{B, E\}}$	12	16	
{B, C, E}	$\chi_5'' = \chi_5' \ \chi_3''^{\downarrow\{A,E, H\}}$		8	
{A, E, H, I}	$\chi_6'' = \chi_6' \ \chi_2''^{\downarrow\{E\}}, \chi_6''^{\downarrow\{A, E,I\}}, \chi_6''^{\downarrow\{H, I\}}$	20	16	
{A, E, G, I}	$\chi_7'' = \chi_7' \ \chi_6''^{\downarrow\{A, E,I\}}, \chi_7''^{\downarrow\{A,G\}}$	12	16	
{A, G, K}	$\chi_8'' = \chi_8' \ \chi_7''^{\downarrow\{A,G\}}$		8	
{H, I, J}	$\chi_9'' = \chi_9' \ \chi_6''^{\downarrow\{H, I\}}$		12	
Computing marginals of singletons:				
{A, F, L}	$\chi_1'^{\downarrow\{L\}}$ (marginal for L)	6		
{A, F, L}	$\chi_1''^{\downarrow\{A\}}$ (marginal for A)	6		
{A, F, L}	$\chi_1''^{\downarrow\{F\}}$ (marginal for F)	6		
{A, D, F, H}	$\chi_2''^{\downarrow\{D\}}$ (marginal for D)	12		
{H, I, J}	$\chi_9''^{\downarrow\{H\}}$ (marginal for H)	10		
{H, I, J}	$\chi_9''^{\downarrow\{I\}}$ (marginal for I)	10		
{H, I, J}	$\chi_9''^{\downarrow\{J\}}$ (marginal for J)	9		
{B, C, E}	$\chi_5''^{\downarrow\{B\}}$ (marginal for B)	6		
{B, C, E}	$\chi_5''^{\downarrow\{C\}}$ (marginal for C)	6		
{B, C, E}	$\chi_5''^{\downarrow\{E\}}$ (marginal for E)	6		
{A, G, K}	$\chi_8''^{\downarrow\{G\}}$ (marginal for G)	6		
{A, G, K}	$\chi_8''^{\downarrow\{K\}}$ (marginal for K)	6		
	TOTALS	221	248	108

Table A.12. Computational Details in the Hugin Architecture for the Stud Farm Problem Using the Junction Tree in Figure A.1.

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{A, F, L}	$\chi_1 = \alpha_L \quad \alpha_F \quad \alpha_A$		16	
{B, C, E}	$\chi_5 = \alpha_C \quad \alpha_E \quad \alpha_B$		16	
{A, G, K}	$\chi_8 = \alpha_K \quad \alpha_E$		8	
{H, I, J}	$\chi_9 = o_J \quad \alpha_J$		12	
Inward propagation (root node is {A, F, L}):				
{H, I, J}	$\chi_9^{\downarrow \{H, I\}}$	8		
{A, G, K}	$\chi_8^{\downarrow \{A, G\}}$	4		
{B, C, E}	$\chi_5^{\downarrow \{B, E\}}$	4		
{A, E, G, I}	$\chi_7' = \alpha_I \quad \chi_8^{\downarrow \{A, G\}}, \chi_7'^{\downarrow \{A, E, I\}}$	8	16	
{A, E, H, I}	$\chi_6' = [\alpha_I \quad \chi_8^{\downarrow \{A, G\}}]^{\downarrow \{A, E, I\}} \quad \chi_9^{\downarrow \{H, I\}}$ $\chi_6'^{\downarrow \{A, E,H\}}$	8	16	
{A, B, D,E}	$\chi_4' = \alpha_D \quad \chi_5^{\downarrow \{B, E\}}, \chi_4'^{\downarrow \{A, D, E\}}$	8	16	
{A, D, E, H}	$\chi_3' = \chi_6'^{\downarrow \{A,E,H\}} \quad \chi_4'^{\downarrow \{A, D, E\}}, \chi_3'^{\downarrow \{A, D, H\}}$	8	16	
{A, D, F, H}	$\chi_2' = \alpha_H \quad \chi_3'^{\downarrow \{A, D, H\}} \chi_2'^{\downarrow \{A, F\}}$	12	16	
{A, F, L}	$\chi_1' = \chi_1 \quad \chi_2'^{\downarrow \{A, F\}} = \chi_1''$		8	
Outward propagation:				
{A, F, L}	$\chi_1''^{\downarrow \{A, F\}}$	4		
AFL-ADFH	$\chi_1''^{\downarrow \{A, F\}} / \chi_2'^{\downarrow \{A, F\}}$			4
{A,D,F,H }	$\chi_2'' = \chi_2' \quad [\chi_1''^{\downarrow \{A, F\}} / \chi_2'^{\downarrow \{A, F\}}]$, $\chi_2''^{\downarrow \{A, D, H\}}$	8	16	
ADFH—ADEH	$\chi_2''^{\downarrow \{A, D, H\}} / \chi_3'^{\downarrow \{A, D, H\}}$			8
{A, D,E,H}	$\chi_3'' = \chi_3' \quad [\chi_2''^{\downarrow \{A, D, H\}} / \chi_3'^{\downarrow \{A, D, H\}}]$, $\chi_3''^{\downarrow \{A, D, E\}}, \chi_3''^{\downarrow \{A,E, H\}}$	16	16	
ADEH-ABDE	$\chi_3''^{\downarrow \{A, D, E\}} / \chi_4'^{\downarrow \{A, D, E\}}$			8
{A, B, D,E}	$\chi_4'' = \chi_4' \quad [\chi_3''^{\downarrow \{A, D, E\}} / \chi_4'^{\downarrow \{A, D, E\}}]$, $\chi_4''^{\downarrow \{B, E\}}$	12	16	
ABDE—BCE	$\chi_4''^{\downarrow \{B, E\}} / \chi_5'^{\downarrow \{B, E\}}$			4
{B, C, E}	$\chi_5'' = \chi_5' \quad [\chi_4''^{\downarrow \{B, E\}} / \chi_5'^{\downarrow \{A, D, E\}}]$		8	

ADEH–AEHI	$\chi_3''^{\downarrow\{A, D, E\}} / \chi_6'^{\downarrow\{A, E, H\}}$			8
{A, E, H, I}	$\chi_6'' = \chi_6' \quad [\chi_3''^{\downarrow\{A, D, E\}} / \chi_6'^{\downarrow\{A, E, H\}}]$, $\chi_6''^{\downarrow\{A, E,I\}}$, $\chi_6''^{\downarrow\{H, I\}}$	20	16	
AEHI—AEGI	$\chi_6''^{\downarrow\{A, E, I\}} / \chi_7'^{\downarrow\{A, E, I\}}$			8
{A, E, G, I}	$\chi_7'' = \chi_7' \quad [\chi_6''^{\downarrow\{A, E, I\}} / \chi_7'^{\downarrow\{A, E, I\}}]$, $\chi_7''^{\downarrow\{A,G\}}$	12	16	
AEGI—AGK	$\chi_7''^{\downarrow\{A,G\}} / \chi_8'^{\downarrow\{A, G\}}$			4
{A, G, K}	$\chi_8'' = \chi_8' \quad [\chi_7''^{\downarrow\{A,G\}} / \chi_8'^{\downarrow\{A, G\}}]$		8	
AEHI—HIJ	$\chi_6''^{\downarrow\{H, I\}} / \chi_9'^{\downarrow\{H, I\}}$			4
{H, I, J}	$\chi_9'' = \chi_9' \quad [\chi_6''^{\downarrow\{H, I\}} / \chi_9'^{\downarrow\{H, I\}}]$		12	
Computing marginals of singletons:				
{A, F, L}	$\chi_1'^{\downarrow\{L\}}$ (marginal for L)	6		
AFL—ADFH	$(\chi_1''^{\downarrow\{A, F\}})^{\downarrow\{A\}}$ (marginal for A)	2		
AFL—ADFH	$(\chi_1''^{\downarrow\{A, F\}})^{\downarrow\{F\}}$ (marginal for F)	2		
ABDE—BCE	$(\chi_4''^{\downarrow\{B, E\}})^{\downarrow\{E\}}$ (marginal for E)	2		
ABDE—BCE	$(\chi_4''^{\downarrow\{B, E\}})^{\downarrow\{B\}}$ (marginal for B)	2		
AEGI—AGK	$(\chi_7''^{\downarrow\{A, G\}})^{\downarrow\{G\}}$ (marginal for G)	2		
AEHI—HIJ	$(\chi_7''^{\downarrow\{H, I\}})^{\downarrow\{H\}}$ (marginal for H)	2		
AEHI—HIJ	$(\chi_7''^{\downarrow\{H, I\}})^{\downarrow\{I\}}$ (marginal for I)	2		
{H, I, J}	$\chi_9''^{\downarrow\{J\}}$ (marginal for J)	9		
{B, C, E}	$\chi_5''^{\downarrow\{C\}}$ (marginal for C)	6		
{A, G, K}	$\chi_8''^{\downarrow\{K\}}$ (marginal for K)	6		
ADFH—ADEH	$(\chi_2''^{\downarrow\{A, D, H\}})^{\downarrow\{D\}}$ (marginal for D)	6		
	TOTALS	179	248	48

Table A.13. Computational Details in the SS Architecture for the Stud Farm Problem Using the Binary Join Tree in Figure A.2.

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
Propagation of Messages:				
{H, I, J}	$\mu^{HIJ \to HI} = (o_J \quad \alpha_J)^{\downarrow \{H, I\}} = \mu^{HI \to AEHI}$	8	12	
{A, G, K}	$\mu^{AGK \to AG} = (\alpha_K \quad \alpha_G)^{\downarrow \{A, G\}} = \mu^{AG \to AEGI}$	4	8	
{A, E, G, I}	$\mu^{AEGI \to AEI} = (\alpha_I \quad \mu^{AG \to AEGI})^{\downarrow \{A, E, I\}} = \mu^{AEI \to AEHI}$	8	16	
{A, E, H, I}	$\mu^{AEHI \to AEH1} = (\mu^{AEI \to AEHI} \quad \mu^{HI \to AEHI})^{\downarrow \{A, E, H\}} = \mu^{AEH1 \to AEH2}$	8	16	
{B, C, E}	$\mu^{BCE \to BE} = (\alpha_C \quad \alpha_E)^{\downarrow \{B, E\}}$	4	8	
{B, E}	$\mu^{BE \to ABDE} = \alpha_B \quad \mu^{BCE \to BE}$		4	
{A, B, D,E}	$\mu^{ABDE \to ADE} = (\alpha_D \quad \mu^{BE \to ABDE})^{\downarrow \{A, D, E\}} = \mu^{ADE \to ADEH}$	8	16	
{A, F, L}	$\mu^{AFL \to AF} = (\alpha_L \quad \alpha_F)^{\downarrow \{A, F\}} = \mu^{AF \to ADFH}$	4	8	
{A,D,F,H }	$\mu^{ADFH \to ADH} = (\alpha_H \quad \mu^{AF \to ADFH})^{\downarrow \{A, D, H\}}$	8	16	
{A, D, E, H}	$\mu^{ADEH \to AEH_2} = (\mu^{ADH \to ADEH} \quad \mu^{ADE \to ADEH})^{\downarrow \{A, E, H\}_2}$	8	16	
$\{A, E, H\}_2$	$\mu^{AEH_2 \to AH} = (\mu^{ADEH \to AEH_2} \quad \mu^{AEH_1 \to AEH_2})^{\downarrow \{A, H\}}$	4	8	
{A, H}	$\mu^{AH \to A} = (\mu^{AEH2 \to AH})^{\downarrow \{A\}}$ $\mu^{AH \to AEH_2} = \alpha_A$	2		
{A}	$\alpha_A \quad \mu^{AH \to A}$ (marginal for A)		2	
$\{A, E, H\}_2$	$\mu^{AEH_2 \to ADEH} = \mu^{AH \to AEH_2} \quad \mu^{AEH_1 \to AEH_2}$ $\mu^{AEH_2 \to AEH_1} = \mu^{AH \to AEH_2} \quad \mu^{ADEH \to AEH_1}$ $= \mu^{AEH_1 \to AEHI}$	-	16	-
{A, D, E, H}	$\mu^{ADEH \to ADH} = (\mu^{AEH_2 \to ADEH} \quad \mu^{ADE \to ADEH})^{\downarrow \{A, D, H\}} = \mu^{ADH \to ADFH}$; $\mu^{ADEH \to ADE} = (\mu^{AEH_2 \to ADEH} \quad \mu^{ADH \to ADEH})^{\downarrow \{A, D, E\}} = \mu^{ADE \to ABDE}$	16	32	-
{A, D, F, H}	$\mu^{ADFH \to AF} = (\alpha_H \quad \mu^{ADH \to ADFH})^{\downarrow \{A, F\}} = \mu^{AF \to AFL}$	12	16	-
(A, F, L}	$\mu^{AFL \to L} = (\alpha_F \quad \mu^{AF \to AFL})^{\downarrow \{L\}}$	6	8	
{L}	$\alpha_L \quad \mu^{AFL \to L}$ (marginal for L)		2	

{A, B, D, E}	$\mu^{ABDE \to BE} = (\alpha_D \ \mu^{ADE \to ABDE})^{\downarrow \{B, E\}}$	12	16	-
{B, E}	$\mu^{BE \to BCE} = \alpha_B \ \mu^{ABDE \to BE}$, $\mu^{BE \to B} = (\alpha_E \ \mu^{ABDE \to BE})^{\downarrow \{B\}}$	2	8	-
{B}	$\mu^{BE \to B} \ \alpha_B$ (marginal for B)		2	
{B, C, E}	$\mu^{BCE \to C} = (\mu^{BE \to BCE} \ \alpha_E)^{\downarrow \{C\}}$	6	8	
{C}	$\alpha_C \ \mu^{BCE \to C}$ (marginal for C)		2	
{A, E, H, I}	$\mu^{AEHI \to AEI} = (\mu^{HI \to AEHI} \ \mu^{AEH1 \to AEHI})^{\downarrow \{A, E, I\}} = \mu^{AEI \to AEGI}$ $\mu^{AEHI \to HI} = (\mu^{AEI \to AEHI} \ \mu^{AEH1 \to AEHI})^{\downarrow \{H, I\}} = \mu^{HI \to HIJ}$	20	32	-
{A, E, G, I}	$\mu^{AEGI \to AG} = (\alpha_I \ \mu^{AEI \to AEGI})^{\downarrow \{A, G\}} = \mu^{AG \to AGK}$	12	16	-
{A, G, K}	$\mu^{AGK \to K} = (\mu^{AG \to AGK} \ \alpha_G)^{\downarrow \{K\}}$	6	8	-
{K}	$\alpha_K \ \mu^{AGK \to K}$ (marginal for K)		2	
{H, I, J}	$\mu^{HIJ \to J} = (\mu^{HI \to HIJ} \ \alpha_J)^{\downarrow \{J\}}$	9	12	-
{ J}	$\mu^{HIJ \to J} \ o_J$ (marginal for J)		3	-
Computing marginals of singletons:				
{A, F}	$(\mu^{AFL \to AF} \ \mu^{ADFH \to AF})^{\downarrow \{F\}}$ (marginal for F)	2	4	
{A, D, E}	$(\mu^{ADEH \to ADE} \ \mu^{ABDE \to ADE})^{\downarrow \{D\}}$(marginal for D)	6	8	
{A, G}	$(\mu^{AEGI \to AG} \ \mu^{AGK \to AG})^{\downarrow \{G\}}$ (marginal for G)	2	4	
{H, I}	$(\mu^{AEHI \to HI} \ \mu^{HIJ \to HI})^{\downarrow \{I\}}$ (marginal for I)	2	4	
$\{A, E, H\}_1$	$(\mu^{AEHI \to AEH_1} \ \mu^{AEH_2 \to AEH_1})^{\downarrow \{E\}}$ (marginal for E)	6	8	
{A, H}	$(\mu^{AEH_2 \to AH} \ \alpha_A)^{\downarrow \{H\}}$ (marginal for H)	2	4	
	TOTALS	187	345	

Table A.14. Computational Details in the LS Architecture for the Genetic Reproduction Problem Using the Junction Tree in Figure A.3.

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{D, E, G}	$\chi_1 = \alpha_D \quad \alpha_G$		27	
{A, A_d}	$\chi_4 = \alpha_A \quad \alpha_{A_d} \quad o_{A_d}$		12	
{B, B_d}	$\chi_5 = \alpha_B \quad \alpha_{B_d} \quad o_{B_d}$		12	
{C, C_d}	$\chi_{10} = \alpha_C \quad \alpha_{C_d} \quad o_{C_d}$		12	
Inward propagation (root node is {D, E, G}):				
{A, A_d}	$\chi_4' = \chi_4 / \chi_4^{\downarrow \{A\}}$	3		6
{B, B_d}	$\chi_5' = \chi_5 / \chi_5^{\downarrow \{B\}}$	3		6
{H, H_d}	$\chi_8' = \chi_8 / \chi_8^{\downarrow \{H\}}$	3		6
{C, C_d}	$\chi_{10}' = \chi_{10} / \chi_{10}^{\downarrow \{C\}}$	3		6
{A, B, E}	$\chi_3' = [\alpha_E \quad \chi_4^{\downarrow \{A\}} \quad \chi_5^{\downarrow \{B\}}] / [\alpha_E \quad \chi_4^{\downarrow \{A\}} \quad \chi_5^{\downarrow \{B\}}]^{\downarrow \{B, E\}}$	18	54	27
{B, C, F}	$\chi_9' = [\alpha_F \quad \chi_{10}^{\downarrow \{C\}}] / [\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}}$	18	27	27
{F, G, H}	$\chi_7' = [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}] / [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}$	18	27	27
{B, F, G}	$\chi_6' = [[\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}} [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}] / [[\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}} [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}]^{\downarrow \{B, G\}}$	18	27	27
{B, E, G}	$\chi_2' = [[[\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}} [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}]^{\downarrow \{B, G\}} [\alpha_E \quad \chi_4^{\downarrow \{A\}} \quad \chi_5^{\downarrow \{B\}}]^{\downarrow \{B, E\}}] / [[[\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}} [\alpha_H \quad \alpha_{Hd}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}]^{\downarrow \{B, G\}} [\alpha_E \quad \chi_4^{\downarrow \{A\}} \quad \chi_5^{\downarrow \{B\}}]^{\downarrow \{B, E\}}]^{\downarrow \{E, G\}}$	18	27	27
{D, E, G}	$\chi_1' = \chi_1 \quad [[[\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}} [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}]^{\downarrow \{B, G\}} [\alpha_E \quad \chi_4^{\downarrow \{A\}} \quad \chi_5^{\downarrow \{B\}}]^{\downarrow \{B, E\}}]^{\downarrow \{E, G\}} = \chi_1''$		27	
Outward propagation:				
{D, E, G}	$\chi_1''^{\downarrow \{E, G\}}$	18		
{B, E, G}	$\chi_2'' = \chi_2' \quad \chi_1''^{\downarrow \{E, G\}}, \chi_2''^{\downarrow \{B, E\}}, \chi_2''^{\downarrow \{B, G\}}$	36	27	
{A, B, E}	$\chi_3'' = \chi_3' \quad \chi_2''^{\downarrow \{B, E\}}, \chi_3''^{\downarrow \{B\}}, \chi_3''^{\downarrow \{A\}}$	48	27	

{A, A_d}	$\chi_4'' = \chi_4' \ \chi_3''^{\downarrow\{A\}}$		6	
{B, B_d}	$\chi_5'' = \chi_5' \ \chi_3''^{\downarrow\{B\}}$		6	
{B, F, G}	$\chi_6'' = \chi_6' \ \chi_2''^{\downarrow\{B,G\}}, \chi_6''^{\downarrow\{B,F\}}, \chi_6''^{\downarrow\{F,G\}}$	36	27	
{F, G, H}	$\chi_7'' = \chi_7' \ \chi_6''^{\downarrow\{F,G\}}, \chi_7''^{\downarrow\{H\}}$	18	27	
{H, H_d}	$\chi_8'' = \chi_8' \ \chi_7''^{\downarrow\{H\}}$		6	
{B, C, F}	$\chi_9'' = \chi_9' \ \chi_6''^{\downarrow\{B,F\}}, \chi_9''^{\downarrow\{C\}}$	18	27	
{C, C_d}	$\chi_{10}'' = \chi_{10}' \ \chi_9''^{\downarrow\{C\}}$		6	
Computing marginals of singletons:				
{D, E, G}	$\chi_1'^{\downarrow\{D\}}$ (marginal for D)	24		
{D, E, G}	$\chi_1'^{\downarrow\{E\}}$ (marginal for E)	24		
{D, E, G}	$\chi_1'^{\downarrow\{G\}}$ (marginal for G)	24		
{B, F, G}	$\chi_6''^{\downarrow\{F\}}$ (marginal for F)	24		
{A, A_d}	$\chi_4''^{\downarrow\{A\}}$ (marginal for A)	3		
{A, A_d}	$\chi_4''^{\downarrow\{A_d\}}$ (marginal for A_d)	4		
{B, B_d}	$\chi_5''^{\downarrow\{B\}}$ (marginal for B)	3		
{B, B_d}	$\chi_5''^{\downarrow\{B_d\}}$ (marginal for B_d)	4		
{C, C_d}	$\chi_{10}''^{\downarrow\{C\}}$ (marginal for C)	3		
{C, C_d}	$\chi_{10}''^{\downarrow\{C_d\}}$ (marginal for C_d)	4		
{H, H_d}	$\chi_8''^{\downarrow\{H\}}$ (marginal for H)	3		
{H, H_d}	$\chi_8''^{\downarrow\{H_d\}}$ (marginal for H_d)	4		
	TOTALS	400	411	159

Table A.15. Computational Details in the Hugin Architecture for the Genetic Reproduction Problem Using the Junction Tree in Figure A.3.

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
At the beginning:				
{D, E, G}	$\chi_1 = \alpha_D \quad \alpha_G$		27	
{A, A_d}	$\chi_4 = \alpha_A \quad \alpha_{A_d} \quad o_{A_d}$		12	
{B, B_d}	$\chi_5 = \alpha_B \quad \alpha_{B_d} \quad o_{B_d}$		12	
{C, C_d}	$\chi_{10} = \alpha_C \quad \alpha_{C_d} \quad o_{C_d}$		12	
Inward propagation (root node is {D, E, G}):				
{A, A_d}	$\chi_4^{\downarrow \{A\}}$	3		
{B, B_d}	$\chi_5^{\downarrow \{B\}}$	3		
{C, C_d}	$\chi_{10}^{\downarrow \{C\}}$	3		
{H, H_d}	$\chi_8^{\downarrow \{H\}}$	3		
{A, B, E}	$\chi_3' = \alpha_E \quad \chi_4^{\downarrow \{A\}} \quad \chi_5^{\downarrow \{B\}}, \chi_3'^{\downarrow \{B, E\}}$	18	54	
{B, C, F}	$\chi_9' = \alpha_F \quad \chi_{10}^{\downarrow \{C\}}, \chi_9'^{\downarrow \{B, F\}}$	18	27	
{F, G, H}	$\chi_7' = \alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}, \chi_7'^{\downarrow \{F, G\}}$	18	27	
{B, F, G}	$\chi_6' = [\alpha_F \quad \chi_{10}^{\downarrow \{C\}}]^{\downarrow \{B, F\}} \quad [\alpha_H \quad \alpha_{H_d}^{\downarrow \{H\}}]^{\downarrow \{F, G\}}, \chi_6'^{\downarrow \{B, G\}}$	18	27	
{B, E, G}	$\chi_2' = \chi_3'^{\downarrow \{B, E\}} \quad \chi_6'^{\downarrow \{B, G\}}, \chi_2'^{\downarrow \{E, G\}}$	18	27	
{D, E, G}	$\chi_1' = \chi_1 \quad \chi_2'^{\downarrow \{E, G\}} = \chi_1''$		27	
Outward propagation:				
{D, E, G}	$\chi_1''^{\downarrow \{E, G\}}$	18		
DEG—BEG	$\chi_1''^{\downarrow \{E, G\}} / \chi_2'^{\downarrow \{E, G\}}$			9
{B, E, G}	$\chi_2'' = \chi_2' \quad [\chi_1''^{\downarrow \{E, G\}} / \chi_2'^{\downarrow \{E, G\}}], \chi_2''^{\downarrow \{B, E\}}, \chi_2''^{\downarrow \{B, G\}}$	36	27	
BEG—ABE	$\chi_2''^{\downarrow \{B, E\}} / \chi_8'^{\downarrow \{B, E\}}$			9
BEG—BFG	$\chi_2''^{\downarrow \{B, G\}} / \chi_6'^{\downarrow \{B, G\}}$			9
{A, B, E}	$\chi_3'' = \chi_3' \quad [\chi_2''^{\downarrow \{B, E\}} / \chi_8'^{\downarrow \{B, E\}}], \chi_3''^{\downarrow \{B\}}$(marg. for B), $\chi_3''^{\downarrow \{A\}}$(marg. for A)	48	27	
ABE—AA_d	$\chi_3''^{\downarrow \{A\}} / \chi_4'$			3
ABE—BB_d	$\chi_3''^{\downarrow \{B\}} / \chi_5'$			3
{A, A_d}	$\chi_4'' = \chi_4' \quad [\chi_3''^{\downarrow \{A\}} / \chi_4']$		6	
{B, B_d}	$\chi_5'' = \chi_5' \quad [\chi_3''^{\downarrow \{B\}} / \chi_5']$		6	

{B, F, G}	$\chi_6'' = \chi_6' \; [\chi_2''^{\downarrow\{B,G\}} / \chi_6'^{\downarrow\{B,G\}}]$, $\chi_6''^{\downarrow\{B,F\}}$, $\chi_6''^{\downarrow\{F,G\}}$	36	27	
BFG—FGH	$\chi_6''^{\downarrow\{F,G\}} / \chi_7'^{\downarrow\{F,G\}}$			9
{F, G, H}	$\chi_7'' = \chi_7' \; [\chi_6''^{\downarrow\{F,G\}} / \chi_7'^{\downarrow\{F,G\}}]$, $\chi_7''^{\downarrow\{H\}}$ (marg. for H)	24	27	
FGH—HH_d	$\chi_7''^{\downarrow\{H\}} / \chi_8'^{\downarrow\{H\}}$			3
{H, H_d}	$\chi_8'' = \chi_8' \; [\chi_7''^{\downarrow\{H\}} / \chi_8'^{\downarrow\{H\}}]$		6	
BFG—BCF	$\chi_6''^{\downarrow\{F,G\}} / \chi_9'^{\downarrow\{B,F\}}$			9
{B, C, F}	$\chi_9'' = \chi_9' \; [\chi_6''^{\downarrow\{F,G\}} / \chi_9'^{\downarrow\{B,F\}}]$, $\chi_9''^{\downarrow\{C\}}$ (marg. for C)	24	27	
BCF—CC_d	$\chi_9''^{\downarrow\{C\}} / \chi_{10}'^{\downarrow\{C\}}$			3
{C, C_d}	$\chi_{10}'' = \chi_{10}' \; [\chi_9''^{\downarrow\{C\}} / \chi_{10}'^{\downarrow\{C\}}]$		6	
Computing marginals of singletons:				
{D, E, G}	$\chi_1'^{\downarrow\{D\}}$ (marginal for D)	24		
DEG—BEG	$[\chi_1'^{\downarrow\{E,G\}}]^{\downarrow\{E\}}$ (marginal for E)	6		
DEG—BEG	$[\chi_1'^{\downarrow\{E,G\}}]^{\downarrow\{G\}}$ (marginal for G)	6		
BFG—BCF	$[\chi_6''^{\downarrow\{B,F\}}]^{\downarrow\{F\}}$ (marginal for F)	6		
{A, A_d}	$\chi_4''^{\downarrow\{A_d\}}$ (marginal for A_d)	4		
{B, B_d}	$\chi_5''^{\downarrow\{B_d\}}$ (marginal for B_d)	4		
{C, C_d}	$\chi_{10}''^{\downarrow\{C_d\}}$ (marginal for C_d)	4		
{H, H_d}	$\chi_8''^{\downarrow\{H_d\}}$ (marginal for H_d)	4		
	TOTALS	346	411	57

Table A.16. Computational Details in the SS Architecture for the Genetic Reproduction Problem Using the Binary Join Tree in Figure A.4.

Computation		# binary arithmetic operations		
At node	Details	+	×	÷
Propagation of Messages:				
$\{C, C_d\}$	$\mu^{CC_d \to C} = (o_{C_d} \quad \alpha_{C_d})^{\downarrow \{C\}}$	3	6	
$\{C\}$	$\mu^{C \to BCF} = \mu^{CC_d \to C} \quad \alpha_C$		3	
$\{B, C, F\}$	$\mu^{BCF \to BF} = (\alpha_F \quad \mu^{C \to BCF})^{\downarrow \{B, F\}} = \mu^{BF \to BFG}$	18	27	
$\{H, H_d\}$	$\mu^{HH_d \to H} = \alpha_{H_d}{}^{\downarrow \{H\}} = \mu^{H \to FGH}$	3		
$\{F, G, H\}$	$\mu^{FGH \to FG} = (\alpha_H \quad \mu^{H \to FGH})^{\downarrow \{F, G\}} = \mu^{FG \to BFG}$	18	27	
$\{B, F, G\}$	$\mu^{BFG \to BG} = (\mu^{BF \to BFG} \quad \mu^{FG \to BFG})^{\downarrow \{B, G\}} = \mu^{BG \to BEG}$	18	27	
$\{D, E, G\}$	$\mu^{DEG \to EG} = (\alpha_D \quad \alpha_G)^{\downarrow \{E, G\}} = \mu^{EG \to BEG}$	18	27	
$\{B, E, G\}$	$\mu^{BEG \to BE_2} = (\mu^{BG \to BEG} \quad \mu^{EG \to BEG})^{\downarrow \{B, E\}}$	18	27	
$\{B, B_d\}$	$\mu^{BB_d \to B} = (o_{B_d} \quad \alpha_{B_d})^{\downarrow \{B\}}$	3	6	
$\{B\}$	$\mu^{B \to BE_1} = \alpha_B \quad \mu^{BB_d \to B}$		3	
$\{A, A_d\}$	$\mu^{AA_d \to A} = (o_{A_d} \quad \alpha_{A_d})^{\downarrow \{A\}}$	3	6	
$\{A\}$	$\mu^{A \to ABE} = \alpha_A \quad \mu^{AA_d \to A}$		3	
$\{A, B, E\}$	$\mu^{ABE \to BE_1} = (\alpha_E \quad \mu^{A \to ABE})^{\downarrow \{B, E\}}$	18	27	
$\{B, E\}_1$	$\mu^{BE_1 \to BE_2} = \mu^{B \to BE_1} \quad \mu^{ABE \to BE_1} = \mu^{BE_2 \to BEG}$		9	
$\{B, E\}_2$	$\mu^{BE_2 \to E} = (\mu^{BEG \to BE_2} \quad \mu^{BE_1 \to BE_2})^{\downarrow \{E\}}$ (marg. for E)	6	9	
$\{B, E, G\}$	$\mu^{BEG \to BG} = (\mu^{BE_2 \to BEG} \quad \mu^{EG \to BEG})^{\downarrow \{B, G\}} = \mu^{BG \to BFG}$	36	54	
	$\mu^{BEG \to EG} = (\mu^{BE2 \to BEG} \quad \mu^{BG \to BEG})^{\downarrow \{E, G\}} = \mu^{EG \to DEG}$			
$\{D, E, G\}$	$\mu^{DEG \to D} = (\alpha_G \quad \mu^{EG \to DEG})^{\downarrow \{D\}}$	24	27	
$\{D\}$	$\alpha_D \quad \mu^{DEG \to D}$ (marg. for D)		3	
$\{B, F, G\}$	$\mu^{BFG \to BF} = (\mu^{FG \to BFG} \quad \mu^{BG \to BFG})^{\downarrow \{B,F\}} = \mu^{BF \to BCG}$	36	54	
	$\mu^{BFG \to FG} = (\mu^{BF \to BFG} \quad \mu^{BG \to BFG})^{\downarrow \{F, G\}} = \mu^{FG \to FGH}$			
$\{B, C, F\}$	$\mu^{BCF \to C} = (\alpha_F \quad \mu^{BF \to BCF})^{\downarrow \{C\}}$	24	27	
$\{C\}$	$\mu^{C \to CC_d} = \mu^{BCF \to C} \quad \alpha_C$		3	
$\{C, C_d\}$	$\mu^{CC_d \to C_d} = (\alpha_{C_d} \quad \mu^{C \to CC_d})^{\downarrow \{C_d\}}$	4	6	
$\{C_d\}$	$o_{C_d} \quad \mu^{CC_d \to C_d}$ (marg. for C_d)		2	
$\{F, G, H\}$	$\mu^{FGH \to H} = (\alpha_H \quad \mu^{FG \to FGH})^{\downarrow \{H\}} = \mu^{H \to HH_d}$	24	27	
$\{H, H_d\}$	$\mu^{HH_d \to H_d} = (\alpha_{H_d} \quad \mu^{H \to HH_d})^{\downarrow \{H_d\}}$ (marg. for H_d)	4	6	
$\{B, E\}_1$	$\mu^{BE_1 \to B} = (\mu^{BE_2 \to BE_1} \quad \mu^{ABE \to BE})^{\downarrow \{B\}}$	6	18	
	$\mu^{BE_1 \to ABE} = \mu^{BE_2 \to BE_1} \quad \mu^{B \to BE_1}$			

{B}	$\mu^{B \to BB_d} = \mu^{BE_1 \to B} \quad \alpha_B$		3
$\{B, B_d\}$	$\mu^{BB_d \to B_d} = (\alpha_{B_d} \quad \mu^{B \to BB_d})^{\downarrow \{B_d\}}$	4	6
$\{B_d\}$	$o_{B_d} \quad \mu^{BB_d \to B_d}$ (marg. for B_d)		2
{A, B, E}	$\mu^{ABE \to A} = (\alpha_E \quad \mu^{BE_1 \to ABE})^{\downarrow \{A\}}$	24	27
{A}	$\mu^{A \to AA_d} = \mu^{ABE \to A} \quad \alpha_A$		3
$\{A, A_d\}$	$\mu^{AA_d \to A_d} = (\alpha_{A_d} \quad \mu^{A \to AA_d})^{\downarrow \{A_d\}}$	4	6
$\{A_d\}$	$o_{A_d} \quad \mu^{AA_d \to A_d}$ (marg. for A_d)		2
Computing marginals of singletons:			
{C}	$\mu^{CC_d \to C} \quad (\alpha_C \quad \mu^{BCF \to C}) = \mu^{CC_d \to C} \quad \mu^{C \to CC_d}$ (marg. for C)		3
{H}	$\mu^{HH_d \to H} \quad \mu^{FGH \to H}$ (marg. for H)		3
{B}	$\mu^{BB_d \to B} \quad (\alpha_B \quad \mu^{BE_1 \to B}) = \mu^{BB_d \to B} \quad \mu^{B \to BB_d}$ (marg. for B)		3
{A}	$\mu^{AA_d \to A} \quad (\alpha_A \quad \mu^{ABE \to A}) = \mu^{AA_d \to A} \quad \mu^{A \to AA_d}$ (marg. for A)		3
{B, F}	$(\mu^{BFG \to BF} \quad \mu^{BCF \to BF})^{\downarrow \{F\}}$ (marg. for F)	6	9
{E, G}	$(\mu^{DEG \to EG} \quad \mu^{BEG \to EG})^{\downarrow \{G\}}$ (marg. for G)	6	9
$\{B, E\}_2$	$(\mu^{BEG \to BE_2} \quad \mu^{BE_1 \to BE_2})^{\downarrow \{E\}}$ (marg. for E)	6	9
	TOTALS	334	522

SELECTED WORKING PAPERS

Unpublished working papers are available via anonymous ftp from
Host: ftp://ftp.bschool.ukans.edu
User ID: (leave blank)
Password: (leave blank)
Directory: /data/pub/pshenoy/
File: wpxxx.ps (put appropriate Working Paper # in place of xxx)

No. 184. "Propagating Belief Functions with Local Computations," Prakash P. Shenoy and Glenn Shafer, February 1986. Appeared in *IEEE Expert*, **1**(3), 1986, 43–52.

No. 190. "Propagating Belief Functions in Qualitative Markov Trees," Glenn Shafer, Prakash P. Shenoy, and Khaled Mellouli, June 1987. Appeared in *International Journal of Approximate Reasoning*, **1**(4), 1987, 349–400.

No. 197. "AUDITOR'S ASSISTANT: A Knowledge Engineering Tool for Audit Decisions," Glenn Shafer, Prakash P. Shenoy, and Rajendra Srivastava, April 1988. Appeared in *Auditing Symposium IX: Proceedings of 1988 Touche Ors/University of Kansas Symposium on Auditing Problems*, 61–84, School of Business, University of Kansas, Lawrence, KS.

No. 200. "Probability Propagation," Glenn Shafer and Prakash P. Shenoy, August 1988. Appeared in *Annals of Mathematics and Artificial Intelligence*, **2**(1–4), 1990, 327–352.

No. 203. "A Valuation-Based Language for Expert Systems," Prakash P. Shenoy, August 1988. Appeared in *International Journal of Approximate Reasoning*, **3**(5), 1989, 383-411.

No. 209. "Axioms for Probability and Belief-Function Propagation," Prakash P. Shenoy and Glenn Shafer, November 1988. Appeared in Shachter, R. D., M. Henrion, L. N. Kanal, and J. F. Lemmer (eds.), *Uncertainty in Artificial Intelligence*, **4**, 1990, 169–198. Reprinted in Shafer, G. and J. Pearl (eds.), *Readings in Uncertain Reasoning*, 1990, 575–610, Morgan Kaufmann, San Mateo, CA.

No. 211. "MacEvidence: A Visual Evidential Language for Knowledge-Based Systems," Yen-Teh Hsia and Prakash P. Shenoy, March 1989. An 8-page summary of this paper appeared as "An evidential language for expert systems," in Ras, Z. W. (ed.), *Methodologies for Intelligent Systems*, **4**, 1989, 9–16, North-Holland, Amsterdam.

No. 213. "On Spohn's Rule for Revision of Beliefs," Prakash P. Shenoy, July 1989. Appeared in *International Journal of Approximate Reasoning*, **5**(2), 1991, 149–181.

No. 216. "Consistency in Valuation-Based Systems," Prakash P. Shenoy, February 1990, revised May 1991. Appeared in *ORSA Journal on Computing*, Vol. 6, No. 3, 1994, 281–291.

No. 220. "Valuation-Based Systems for Bayesian Decision Analysis," Prakash P. Shenoy, April 1990, revised May 1991. Appeared in *Operations Research*, **40**(3), 1992, 463–484.

No. 221. "Valuation-Based Systems for Discrete Optimization," Prakash P. Shenoy, June 1990. Appeared in Bonissone, P. P., M. Henrion, L. N. Kanal, and J. F. Lemmer, eds., *Uncertainty in Artificial Intelligence*, **6**, 1991, 385–400, North-Holland, Amsterdam.

No. 223. "A New Method for Representing and Solving Bayesian Decision Problems," Prakash P. Shenoy, September 1990. Appeared in: Hand, D. J. (ed.), *Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, 1993, 119–138, Chapman & Hall, London, England.

No. 226. "Valuation-Based Systems: A Framework for Managing Uncertainty in Expert Systems," Prakash P. Shenoy, March, 1991. Appeared in: Zadeh, L. A. and J. Kacprzyk (eds.), *Fuzzy Logic for the Management of Uncertainty*, 1992, 83–104, John Wiley and Sons, New York, NY.

No. 227. "Valuation Networks, Decision Trees, and Influence Diagrams: A Comparison," Prakash

P. Shenoy, June 1991. Appeared as: "A Comparison of Graphical Techniques for Decision Analysis" in *European Journal of Operational Research*, Vol. 78, No. 1, 1994, 1–21.

No. 233. "Using Possibility Theory in Expert Systems," Prakash P. Shenoy, September 1991. Appeared in *Fuzzy Sets and Systems*, **52**(2), 1992, 129–142.

No. 236. "Conditional Independence in Valuation-Based Systems," Prakash P. Shenoy, September 1991. Appeared in *International Journal of Approximate Reasoning*, **10**(3), 1994, 203–234.

No. 238. "Valuation Networks and Conditional Independence," Prakash P. Shenoy, September 1992. Appeared as "Representing Conditional Independence Relations by Valuation Networks" in *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **2**(2), 1994, 143–165.

No. 239. "Game Trees for Decision Analysis," Prakash P. Shenoy, February 1993. Revised February 1994. An 8-page summary titled "Information Sets in Decision Theory" appeared in Clarke, M., R. Kruse and S. Moral (eds.), *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, Lecture Notes in Computer Science No. 747, 1993, 318–325, Springer-Verlag, Berlin.

No. 242. "A Theory of Coarse Utility," Liping Liu and Prakash P. Shenoy, February 1993. Revised September 1993. Appeared in *Journal of Risk and Uncertainty*, Vol. 11, 1995, pp. 17–49.

No. 245. "Modeling Ignorance in Uncertainty Theories," Prakash P. Shenoy, April 1993. Appeared in Gammerman, A. (ed.), *Probabilistic Reasoning and Bayesian Belief Networks*, 1995, 71–96, Alfred Waller, Henley-on-Thames, UK.

No. 246. "Valuation Network Representation and Solution of Asymmetric Decision Problems," Prakash P. Shenoy, April 1993. Revised September 1995. A 10-page summary of this paper appeared as "Representing and Solving Asymmetric Decision problems Using Valuation Networks" in Fisher, D. and H.-J. Lenz (eds.), *Artificial Intelligence and Statistics V*, Lecture Notes in Statistics, **112**, 99–108, Springer-Verlag, New York, 1996.

No. 247. "Inducing Attitude Formation Models Using TETRAD," Sanjay Mishra and Prakash P. Shenoy, May 1993. Revised October 1993. Appeared as "Attitude Formation Models: Insights from TETRAD" in Cheeseman, P. and R. W. Oldford (eds.), *Selecting Models from Data: Artificial Intelligence and Statistics IV*, Lecture Notes in Statistics No. 89, 1994, 223–232, Springer-Verlag, Berlin.

No. 258. "A Note on Kirkwood's Algebraic Method for Decision Problems," Rui Guo and Prakash P. Shenoy, November 1993. Revised May 1994. To appear in *European Journal of Operational Research*, 1996.

No. 261. "A New Pruning Method for Solving Decision Trees and Game Trees," Prakash P. Shenoy, March 1994. Appeared in: Besnard, P. And S. Hanks (eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*, 1995, 482–490, Morgan Kaufmann, San Francisco, CA.

No. 267. "Computing Marginals Using Local Computation," Steffen L. Lauritzen and Prakash P. Shenoy, July 1995, revised May 1996.

No. 270. "Binary Join Trees for Computing Marginals in the Shenoy-Shafer Architecture," Prakash P. Shenoy, December 1995. An 8-pp summary titled "Binary Join Trees" appeared in: Horvitz, E. and F. V. Jensen (eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Twelfth Conference*, 1996, 492–499, Morgan Kaufmann, San Francisco, CA.

No. 271. "A Comparison of Graphical Techniques for Asymmetric Decision Problems," Concha Bielza and Prakash P. Shenoy, February 1996, revised June 1996.

No. 273. "A Forward Monte Carlo Method for Solving Influence Diagrams Using Local Computation," John M. Charnes and Prakash P. Shenoy, February 1996.