AN ABSTRACT OF THE THESIS OF

Mohammad Borujerdi for the degree of Doctor of Philosophy in Computer Science presented on April 6, 1999.

Title: Optimal Inference with Local Expressions.

Abstract approved: _____

Bruce D'Ambrosio

Probabilistic inference using Bayesian networks is now a well-established approach for reasoning under uncertainty. Among many efficiency-driven techniques which have been developed, the Optimal Factoring Problem (OFP) is distinguished for presenting a combinatorial optimization point of view on the problem.

The contribution of this thesis is to extend OFP into a theoretical framework that not only covers the standard Bayesian networks but also includes non-standard Bayesian networks. A non-standard Bayesian network has structures within its local distributions that are significant to the problem. This thesis presents *value sets algebra* as a coherent framework that facilitates formal treatments of inference in both standard and non-standard Bayesian networks as a combinatorial optimization problem.

Parallel to value sets algebra theory *local expression languages* allow one to symbolically encode Bayesian network distributions. Such symbolic encodings allow all the structural and numerical information in distributions to be represented in the most compact form. However, the symbolic and syntactic

flexibilities in local expression languages have the usual drawback of allowing possible incoherent expressions. Value sets algebra leads us to an efficient coherency verification on such expressions.

This thesis views optimal inference with local expressions as an optimal search problem. The search space for this problem is shown to be so large that it renders any exhaustive search impractical. Hence it is necessary to turn to heuristic solutions. Using A* heuristic framework and ideas from OFP, which is the counterpart of this problem for standard Bayesian networks, a heuristic algorithm for the problem is developed. As a key feature, this algorithm differentiates between symbolic combinations of expressions and arithmetic operations in the expressions. Cost bearing arithmetic operations are performed only when sufficient information is available to guarantee that no saving opportunities are lost. On the other hand, expressions are combined in a way that quickly provides maximum opportunity for efficient arithmetic operations.

This thesis also explores the representation of Intercausal Independencies (ICI) in Bayesian networks and defines some new operators in local expression language which are shown to facilitate more efficient ICI representations.

Optimal Inference with Local Expressions

by

Mohammad Borujerdi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed April 6, 1999
Commencement June 1999

Doctor of Philosophy thesis of Mohammad Borujerdi presented on April 6, 1999

APPROVED:

Major Professor, representing Computer Science

Head of Department of Computer Science

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Mohammad Borujerdi, Author

Approved by Committee:

Major Professor (Bruce D'Ambrosio)

Committee Member (Michael Quinn)

Committee Member (Timothy Budd)

Committee Member (Prasad Tadepalli)

Graduate School Representative (Jonathan King)

Date thesis presented April 6, 1999

ACKNOWLEDGMENT

I would like to express my gratitude to my major professor and advisor Professor Bruce D'Ambrosio for all the help and support that he gave me throughout this work. I would also like to thank other wonderful members of my committee, Professors Michael Quinn, Prasad Tadepalli, Timothy Budd, and Jonathan King for their assistance.

Many thanks to Computer Science Dept. faculty, staff, and support, especially to Professor Toshimi Minoura and Mrs. Bernie Feyerherm.

I would like to thank my Mother for letting me come here to pursue a dream and I cannot find enough words to thank my dear wife - Nasrin whose understanding and patience made this possible.

TABLE OF CONTENTS

Page

Chapter 1: Introduction **1**

1.1 Motivation 1

1.2 Background 2

1.3 Overview 3

Chapter 2: Local Expressions **4**

Chapter 3: Value Sets Algebra **11**

3.1 Theory of Value Sets Algebra 12

3.2 Value Sets Join Chain Optimization Problem 20

3.3 Inference in Bayesian Networks and Value Sets Algebra 21

Chapter 4: Coherency of Local Expressions **23**

4.1 Canonical Form 23

4.2 Coherency Verification Procedure 24

Chapter 5: Local Expressions Inference **28**

5.1 Introduction 28

5.2 Local Expressions Inference as an Optimal Search Problem . . . 29

5.3 A Review of Optimal Factoring Problem and its Heuristic Solution 32

5.4 SPI*: A Heuristic Algorithm for Local Expressions Inference . . 35

5.4.1 Adapting the Remaining Size Criterion 35

5.4.2 Performing the Operations in Expressions 36

5.4.2.1 Marginalization Related (MGR) Operations . . . 36

5.4.2.2 Subset-Dimension Related (SDR) Operations . . 38

5.4.2.3 Term Summation Related (TSR) Operations . . . 38

TABLE OF CONTENTS (Continued)

Page

5.5 An Example Execution of SPI* Algorithm 40

5.6 Performance of the Heuristic 43

5.6.1 Asymmetries and Contingencies 43

5.6.2 Intercausal Independencies 50

5.6.3 Complexity 51

Chapter 6: Intercausal Independencies in Bayesian Networks 53

6.1 Heterogeneous Factorization 54

6.2 Temporal Transformation 56

6.3 A Look at the Performance of the Approaches 57

6.4 A Multiplicatively Factored Representation 61

6.5 New Representation and Inference Algorithms 64

Chapter 7: Conclusions and Future Work 69

7.1 Conclusions 69

7.2 Future Work 70

Bibliography 71

LIST OF FIGURES

Figure		Page
2.1	An example of noisy-or interaction.	6
5.1	Part of the search space for finding the optimal order of operations	32
5.2	An example network for the heuristic algorithm.	40
5.3	Influence diagram for the oil wildcatter example.	44
5.4	Influence diagram for the car buyer example.	46
5.5	Influence diagram for the Generalized Buying Problem.	47
5.6	Savings and Number of decisions graph for Generalized Buying Problem.	48
5.7	Savings and savings factor graph for Generalized Buying Problem.	48
5.8	An Example CPCS-type Bayesian network.	51
6.1	A Bayesian network for studying heterogeneous factorization.	54
6.2	Deputation Bayesian network for Fig. 6.1 network.	55
6.3	A Bayesian network for a multiple cause situation.	56
6.4	A temporal transformation of the network in Fig. 6.3.	57
6.5	An example network for temporal transformation.	59
6.6	Another example network for temporal transformation.	60
6.7	An example BN2O network.	66

LIST OF TABLES

Table		Page
2.1	Evaluation for the noisy-or expression of Fig. 1	8
5.1	Savings vs the number of asymmetric decisions.	49
5.2	Savings vs the savings factor.	49
6.1	Characteristic terms for the expressions in example BN2O network	67

OPTIMAL INFERENCE WITH LOCAL EXPRESSIONS

Chapter 1

INTRODUCTION

1.1 Motivation

A Bayesian network representation of a probabilistic situation consists of a graphical structure and a set of conditional and marginal distributions over the variables involved. The graphical level, which represents the conditional independencies, is used by inference algorithms to reduce the computational complexity of inference. This is good in general, but in many situations there are interesting structural properties that neither the graphical level nor the ordinary joint conditional distributions of Bayesian networks can effectively represent. The local expression language provides an efficient representation mechanism for such situations and thus can enhance and improve probabilistic inference.

Intercausal independencies (ICI) and asymmetric models are two classes of such situations which have been applied to a range of applications including diagnosing multiple diseases and decision making. The distinguishing feature of the local expression approach in comparison with others is the explicit use of symbolic manipulation to simplify the inevitable numerical calculations in probability evaluation. Since probability distributions are algebraic objects, our findings will potentially contribute to the field of symbolic algebra as well.

1.2 Background

A Bayesian network represents a full joint probability distribution over a set of variables in the form a directed acyclic graph (DAG) and a set of local distributions. If such a DAG is sparse, local distributions will be small and probabilistic computations with this representation will involve fewer numbers and will be more efficient than computations with the full joint distribution. The sparseness of the DAG follows from conditional independencies among the variables which are represented as nodes.

During the early stages of the development of Bayesian networks emphasis was largely on decomposing the large joint probability distributions. Using the DAGs as graphical representations and smaller local distributions, such decompositions eased knowledge elicitation and inference computations (Pearl, 1988). In that context local expressions were used in the nodes of Bayesian network DAGs to represent the joint conditional probability distribution (JCD) of the node variable given its parents. We refer to such models and their corresponding local expressions as standard Bayesian networks and standard local expressions.

Later, local distribution of Bayesian networks started to involve finer grain representations with even more potential to improve the efficiency of the inference computations. Such developments include noisy-or (Pearl, 1988), (Srinivas, 1993) , contingencies (Fung R., & Shachter R., 1991), additive models (Dagum & Galper, 1993), similarity networks (Heckerman, 1991), and the structured distributions (Boutilier, Friedman, Goldszmidt, & Koller, 1996). Along with those developments local expressions were extended and a language with a formal grammar was defined for them. This allowed local expressions

to (D'Ambrosio, 1995) represent more dependencies including intercausal and asymmetric dependencies. The extended local expressions brought some interesting issues with them, including the coherency of local expressions and how to do inference with them.

1.3 Overview

Chapter Two is an extensive review of local expression languages including their grammar, representation, and examples.

In Chapter Three "value sets algebra," a theoretical framework for operation on local expressions, is presented. Using value sets algebra a combinatorial optimization problem is defined for joining value sets. Then it is shown that probabilistic inference in Bayesian networks with extended local expressions can be described as instances of that optimization problem.

Chapter Four is about the coherency of local expressions. It is shown how the symbolic representation of local expressions can be verified to be consistent with the numerical constraints of axiomatic probability theory.

In Chapter Five solutions to the optimal inference with local expressions are studied. Optimal factoring, which is a closely related problem, is reviewed and some additional insights are provided which eventually lead to a heuristic algorithm for the problem of optimal inference with local expressions.

Chapter Six focuses on the representation of intercausal independencies in Bayesian networks. Major approaches for representing independencies among causal effects in Bayesian networks are compared. Some of the factors which influence the performance of the approaches in different situations are identified. Finally a new and better representation for intercausal independencies as local expressions is introduced.

Chapter 2

LOCAL EXPRESSIONS

A local expression language is an extension to the standard representation for Bayesian networks. This extended expression language is useful for compact representation of various models for interaction among antecedents.

The intuition behind local expressions is simple. Any marginal or conditional distribution in Bayesian networks can be written as an algebraic expression of distributions whose state-spaces are subsets of the original. For example, the marginal distribution $P(A)$ (assuming $P(A = t) = 0.7$, and $P(A = f) = 0.3$) can be written as

$$A_t + A_f$$

where A_t, for example, has the value 0.7 and is defined over $A = t$. But such a representation, although complete, is not always beneficial, since its space and inference time complexity are exponential in the number of antecedent nodes.

However, there are situations where additional independencies in the local interactions exist, and such independencies can be exploited to write more compact local expressions and obtain faster inference time. In the rest of this chapter we will first define the syntax and semantics for local expressions. Then we will review some of those interaction models to see how local expressions can capture the independencies and other structural properties in the models.

The formal syntax for our expression language is shown below. Note how terms eventually reduce to one or more *distributions*. Distributions are defined

over some subset of the Cartesian product of domains of their conditioned and conditioning variables. In our presentation the actual numeric values in the distributions will be ignored, since they are not germane to the discussion. Also, our actual syntax uses prefix notation; however, for readability we will use infix notation throughout the thesis.

$$
\begin{aligned}
exp &\rightarrow term \mid (+\ term\ term\text{-}set) \mid \\
&\rightarrow (-\ term\ term\text{-}set) \mid \\
&\rightarrow (*\ term\ term\text{-}set) \\
term &\rightarrow exp \mid distribution. \\
term\text{-}set &\rightarrow term \mid term\ term\text{-}set. \\
distribution &\rightarrow name_{dimensions}. \\
dimensions &\rightarrow conditioned \mid \\
&\rightarrow conditioned\ \text{“}\mid\text{”}\ \ conditioning. \\
conditioned &\rightarrow var\text{-}name_{domain} \\
&\rightarrow var\text{-}name_{domain} conditioned. \\
conditioning &\rightarrow var\text{-}name_{domain} \\
&\rightarrow var\text{-}name_{domain} \mid conditioning. \\
domain &\rightarrow \text{“ ”} \mid value \mid value\text{-}set. \\
value\text{-}set &\rightarrow value \mid value,\ value\text{-}set.
\end{aligned}
$$

For local expressions we define their semantics in the following evaluation rule, and then we move on to consider the use of the local expression language to represent several commonly occurring intra-distribution structures. The evaluation rule is as follows.

> An expression is equivalent to the distribution obtained by evaluating it using the standard rules of algebra for each possible combination of antecedent values.

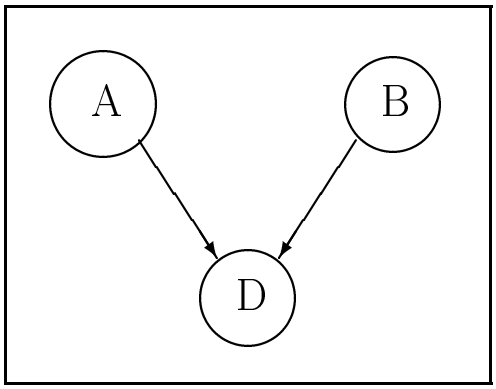

FIGURE 2.1: An example of noisy-or interaction.

Noisy-or If A and B have a noisy-or influence on D in the network shown in Fig. 2.1 the following expression can be written to represent the dependence of D on A and B, following Pearl (Pearl, 1988):

$$\begin{aligned} P(D=t) &= 1-(1-c_A(D))(1-c_B(D)) \\ P(D=f) &= (1-c_A(D))(1-c_B(D)) \end{aligned}$$

where $c_A(D)$ is the probability that D is true given that A is true and B is false. The notation can be made slightly more compact by defining

$$c'_A(D) = (1-c_A(D))$$

where

$$\begin{aligned} c'_A(D) &= 1-c_A(D), A=t \\ &= 1, A=f \end{aligned}$$

Now $c'_A(D)$ is a pair of numbers expressing the dependence of $P(D)$ on A whereas, $c_A(D)$ is a single number ($P(D = t|A = t, B = f)$). The expression can now be rewritten as:

$$\begin{aligned} P(D = t) &= 1 - c'_A(D) * c'_B(D) \\ P(D = f) &= c'_A(D) * c'_B(D) \end{aligned}$$

The size of this expression is linear in the number of antecedents, it captures the structure of the interaction, and, as Heckerman showed (Heckerman, 1989), it can be manually manipulated to perform efficient inference.

$$e(D) = 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}}$$

Note the two instances of $c'_A(D)$ in the expression where the numeric distributions for the instances are identical, their defining state-spaces are different.

Having shown that a noisy-or can be expressed in our syntax, we will next examine whether the expression semantically matches those standardly attributed to the noisy-or structural model. Performing the evaluation in the expressions semantic rule for our simple example yields the Table 2.1 which is, in fact, exactly the standard semantics attributed to noisy-or.

Asymmetries (Geiger D. & Heckerman D., 1991) pointed out that probabilistic relationships are often asymmetric. An example asymmetric relationship was presented by Geiger and Heckerman in which, a variable *Badge* (B) depends on a second variable, *Hypothesis* (H). *Badge* also depends on a third variable, *Gender* (G), but only when *Hypothesis* is either "worker" or "executive" (*Hypothesis* takes four values, "worker" (w), "executive" (e) , "visitor"

A	B	D	
		t	f
t	t	1-(1-$c_D(A)$)*(1-$c_D(B)$)	(1-$c_D(A)$)*(1-$c_D(B)$)
t	f	1-(1-$c_D(A)$)*(1)	(1-$c_D(A)$)*(1)
f	t	1-(1)*(1-$c_D(B)$)	(1)*(1-$c_D(B)$)
f	f	1-(1)*(1)	(1)*(1)

TABLE 2.1: Evaluation for the noisy-or expression of Fig. 1

(v), and "spy" (s)). The following expression can be written for this structure.

$$exp(Badge) = P_{B_{t,f}|H_{s,v}} + P_{B_{t,f}|H_{w,e},G_{m,f}}$$

Contingencies Contingency is a form of asymmetry in which a variable's existence is contingent on other variables. For example, W might only exist when $X = t$, and it might depend on the values of variables Y and Z (Fung R., & Shachter R., 1991):

$$exp(W) = P_{W_{t,f}|X_t Y_{t,f} Z_{t,f}} + P_{W_\emptyset|X_f}$$

The second term in the above expression represents the nonexistence of W and ensures that we are able to recover the joint probability across all the variables by forming the product of all local expressions. Another way to ensure this is to make it a requirement that every local expression must be defined for every instantiation of its parent set. Furthermore, for each such instantiation, the local expression must describe the distribution of the mass for that instantiation. The second term in the above expression represents a fact about the mass corresponding to specific instantiations of the parents (all

those in which $X = f$). It shows that the mass is not assigned to any of the values in the domain of the variable W. However, this mass must be explicitly represented so that this expression can be combined with others to properly recover the full joint.

Additive Decompositions The utility of additive decomposition of conditional distributions has been noted by Dagum and Galper (Dagum & Galper, 1993). Local expressions can easily represent the corresponding model:

$$exp(Y) = a_1 * p_{Y_{t,f}|X1_{t,f}} + a_2 * p_{Y|X2_{t,f}}$$

With regard to this example we must mention that in the local expression language domains are not restricted to $\{t, f\}$; we use that domain merely for concreteness. Also, it is possible to decompose into conditioning subsets rather than individual variables as Dagum and Galper described.

Inference computation with local expressions can be done with an algorithm which is an extension of set factoring algorithm (Li & D'Ambrosio, 1994). We will refer to this algorithm as ESPI hereafter. ESPI algorithm tries to find the best order of operations on local expressions by successively selecting and combining candidate expressions whose combination will have the smallest set of remaining variables (D'Ambrosio, 1995). Compared to the set factoring algorithm, the extended algorithm has a more comprehensive combination candidate generation procedure. It also includes heuristic adjustments to the scoring function for combination candidates. For example, the number of terms in an expression is taken into account in scoring. After the candidate is selected, the combined expression is immediately reduced to a distribution by performing the arithmetic operations in the expression. For example, consider the network

in Fig. 2.1 which involves the following local expressions.

$$\begin{aligned} e(D) &= 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}} \\ e(A) &= A_t + A_f \\ e(B) &= B_t + B_f \end{aligned}$$

Combining expressions for A and D will give,

$$\begin{aligned} e(AD) &= 1_{D_t} * A_t - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} * A_t \\ &\quad + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}} * A_t \\ &\quad 1_{D_t} * A_f - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} * A_f \\ &\quad + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}} * A_f \end{aligned}$$

The algorithm at this point performs all the relevant products inside the expression and essentially reduces the expression to a distribution. Later in chapter five we will present a new way for extending the set factoring heuristic to inference with these expressions. Also in chapter six the performance of this approach will be analyzed along with other prominent approaches for representation of noisy-or situations.

Chapter 3

VALUE SETS ALGEBRA

Probability distributions are algebraic objects at the center of a highly developed scientific research area called "Probabilistic Reasoning". During the early stages of the development of that area, emphasis was mainly on capturing value independent conditional independence to decompose the representation of large joint probability distributions to ease knowledge elicitation and inference computations (Pearl, 1988). Symbolic Probabilistic Inference (SPI) (D'Ambrosio, 1990) approached the efficient probabilistic inference in an algebraic way that later led to a combinatorial optimization point of view on the same problem known as Optimal Factoring Problem (OFP) (Li & D'Ambrosio, 1994).

Nowadays decomposition efforts in the same area have reached the distributions at local levels. Researchers (Geiger D. & Heckerman D., 1991); (Srinivas, 1993); (Heckerman & Breese, 1994); (Zhang & Poole, 1996) have developed local distribution models that involve finer grain decompositions. These models are different from standard distributions and they can potentially improve the efficiency of the inference computations even further. OFP framework made it possible to define the inference in standard Bayesian networks as a combinatorial optimization problem. Now it is our objective to develop a theoretical framework for non-standard Bayesian networks where there are structures within local distributions and they are significant to the problem. Our goal is to provide a coherent set of definitions that can facilitate a formal treatment

of probabilistic inference in non-standard Bayesian network as a combinatorial optimization problem.

3.1 Theory of Value Sets Algebra

Let us first introduce some notation. Throughout this section first uppercase letters of the alphabet (e.g., A, B, C) denote propositional variables. The last uppercase alphabet letters (e.g., X, Y, Z) are used to represent sets of variables. A subscripted variable denotes a set of specific instances or values of that variable as indicated in the subscript. The domain of a variable A, which is the finite set of all instances of that variable, is denoted by d(A). Super domain of a variable A is denoted by SD(A) and is the power set of the domain of that variable, i.e., SD(A) = $\{d\colon\ d \subseteq \mathrm{d}(A)\ \}$.

The domain of a set of variables $V = \{V^1, V^2, \ldots, V^n\ \}$, is the Cartesian product of the domains of the variables in the set, i.e., $d(V) = d(V^1) \times d(V^2) \times \ldots \times d(V^n)$. Similarly, the super domain of a set of variables is the Cartesian product of the super domains of all the variables in the set, i.e., $SD(V) = SD(V^1) \times SD(V^2) \times \ldots \times SD(V^n)$.

Definition 1. Simple Super domain of a variable A, denoted by D(A), is the subset of SD(A) where the size of every element is less than two, i.e., D(A) = $\{sd\colon\ sd \in \mathrm{SD}(A)$ and $|sd| < 2\ \}$. The simple super domain of a set of variables is the Cartesian product of the simple super domains of all the variables in the set, i.e., $D(V) = D(V^1) \times D(V^2) \times \ldots \times D(V^n)$. An element of $D(V)$ is called a *point* and it is an n-tuple $(s_1, s_2, \ldots, s_n)$ where $s_i \in D(V^i)$. Given an $n-tuple\ s = (s_1, s_2, \ldots, s_n)$ we denote its component corresponding to V^i as $s(V^i)$. Two points $s1$ and $s2$ are equal if both correspond to the same set of variables V and $\forall v \in V, s1(v) = s2(v)$.

Definition 2. Subsets of $D(V)$ are called point sets. Dimension of a point set $S \subset D(V)$ is denoted by $dim(S)$, and it is the set V. The scope of V^i in S is denoted by $S(V^i) = I \subseteq D(V^i)$ and it is the set of those elements of $D(V^i)$ which are present in at least one element of S. In other words, $I = \{v : v \in D(V^i), \exists s \in S, s(V^i) = v\}$. We use $S(V^k) = \emptyset$ to represent that $V^k \notin dim(S)$ or V^k has an empty scope in S. On the other hand if $S(V^i) = D(V^i) \backslash \{\emptyset\}$ we say V^i has a full scope in S.

Note that if $S(V^i) = \{\emptyset\}$ then $V^i \in dim(S)$ and V^i has a nonempty scope in S.

Definition 3. The dimension of a point set can be reduced by reducing the scopes of some variables to empty set. Suppose $S \subset D(V)$ and $dim(S) = V$. We denote the reduction of the dimension of S from V to Y by $\propto_{-Z} S$ where $Z = V - Y$. Such a reduction results in $\hat{S}_Y$ where $Y \subset V$ and $\forall s \in \hat{S}_Y$, if $A \in Z$ then $s(A)$ does not exist.

Definition 4. The dimension of a point set can be expanded by adding some variables with their full scope to the dimension of that point set. Suppose $S \subset D(V)$ and $dim(S) = V$. We denote the expansion of the dimension of S from V to Y by $\propto_{+Z} S$ where $Z = Y - V$. Such an expansion results in $T_Y = S \times (fd(Z^1) \times fd(Z^2) \ldots \times fd(Z^n))$ where $Z^i \in Z$ for $i = 1, 2, \ldots n$ and $fd(Z^i) = D(Z^i) \backslash \{\emptyset\}$.

Union, set difference, and intersection for sets of points with the same dimensions are the same operations defined for sets in general. Using definition 4 one only needs to expand the dimensions of operands to the union of their dimensions before performing such operations on sets of points with different dimensions. Cartesian product of sets of points is the same as the Cartesian product for sets in general. In the Cartesian product of two sets of points such

as S_1 and S_2 denoted by $S_1 \times S_2$, the product of two points such as s_1 and s_2 is denoted by (s_1, s_2).

Definition 5. Join of two sets of points S_1 and S_2, is denoted by $S_1 \bowtie S_2$.

$$\begin{aligned} S_1 \bowtie S_2 &= \{(s_1, s_2) | s_1 \in S_1, s_2 \in S_2, \\ &\quad \forall v \text{ if } \exists s_1(v) \text{ and } \exists s_2(v),\ s_1(v) = s_2(v)\} \end{aligned}$$

So join produces a subset of Cartesian product where every variable has only one mapping at every point.

Definition 6. Full join of two point sets $S_1 \bowtie_F S_2$ is defined as follows. Suppose $Y = dim(S_1)$ and $Z = dim(S_2)$. First expand the dimensions of S_1 and S_2 to $V = Y \cup Z$ to get S_1^V and S_2^V.

$$\begin{aligned} S_1 \bowtie_F S_2 &= \{(s_1, s_2) | s_1 \in S_1^V, s_2 \in S_2^V, \\ &\quad \forall v,\ s_1(v) = s_2(v)\} \end{aligned}$$

Lemma 0. Join and Full Join are equivalent.

Proof. Consider point sets S_1 and S_2 where $Y = dim(S_1)$, $Z = dim(S_2)$, $V = Y \cup Z$, and $I = Y \cap Z$. We must prove that $J = FJ$ where $J = S_1 \bowtie S_2$ and $FJ = S_1 \bowtie_F S_2$.

We can describe V as the union of three disjoint sets. $V = I \cup (Y - Z) \cup (Z - Y)$. Suppose $(s_1, s_2) \in J$ then,

$$\begin{aligned} \forall v \in V \quad &\text{if } v \in I && \text{then } (s_1, s_2)(v) = s_1(v) = s_2(v) \\ &\text{if } v \in (Y - Z) && \text{then } (s_1, s_2)(v) = s_1(v) \\ &\text{if } v \in (Z - Y) && \text{then } (s_1, s_2)(v) = s_2(v) \end{aligned}$$

Now let $s_k^R, k = 1, 2, R = I, Y - Z, Z - Y$ denote the elements of the s_k n-tuple which correspond to variables in R. For example, s_2^{Z-Y} are the components in s_2 which correspond to variables that are exclusively in dimension of

S_2 and not in dimension of S_1. We form two new points s'_1 and s'_2 by forming products of s_1 and s_2 in the following way: $s'_1 = (s_1, s_2^{Z-Y})$ and $s'_2 = (s_2, s_1^{Y-Z})$. By definition 6 we know that $(s'_1, s'_2) \in FJ$. Furthermore,

$$\begin{aligned}
\forall v \in V \quad \text{if } v \in I \qquad \text{then } (s'_1, s'_2)(v) &= s'_1(v) = s'_2(v) \\
&= s_1(v) = s_2(v) \\
&= (s_1, s_2)(v) \\
\text{if } v \in (Y - Z) \quad \text{then } (s'_1, s'_2)(v) &= s'_1(v) = s_1(v) \\
&= (s_1, s_2)(v) \\
\text{if } v \in (Z - Y) \quad \text{then } (s'_1, s'_2)(v) &= s'_2(v) = s_2(v) \\
&= (s_1, s_2)(v)
\end{aligned}$$

So $\forall v \in V, (s_1, s_2)(v) = (s'_1, s'_2)(v)$ and $(s_1, s_2)(v) \in FJ$.

Now suppose $(s_1, s_2) \in FJ$. Then $\forall v \in V$ we have $s_1(v) = s_2(v)$. We form two new points $s"_1$ and $s"_2$ by removing parts of s_1 and s_2 respectively in the following way: $s"_1 = (s_1^I, s_1^{Y-Z})$ and $s"_2 = (s_2^I, s_2^{Z-Y})$. By definition 5 we know that $(s"_1, s"_2) \in J$. Furthermore,

$$\begin{aligned}
\forall v \in V \quad \text{if } v \in I \qquad \text{then } (s"_1, s"_2)(v) &= s"_1(v) = s"_2(v) \\
&= s_1(v) = s_2(v) \\
&= (s_1, s_2)(v) \\
\text{if } v \in (Y - Z) \quad \text{then } (s"_1, s"_2)(v) &= s"_1(v) = s_1(v) \\
&= (s_1, s_2)(v) \\
\text{if } v \in (Z - Y) \quad \text{then } (s"_1, s"_2)(v) &= s"_2(v) = s_2(v) \\
&= (s_1, s_2)(v)
\end{aligned}$$

So $\forall v \in V, (s_1, s_2)(v) = (s"_1, s"_2)(v)$ and $(s_1, s_2)(v) \in J$. Hence $J = FJ$.

Definition 7. A value set f_S is a mapping between the point set S and the set of real numbers. Such a mapping assigns a real number value to every

point in S. The size of a value set is the size of its domain point set. A subset of a value set f_S is another value set $\hat{f}_R$ where $R \subset S$ and $\forall r \in R, f_r = \hat{f}_r$.

Definition 8. A dimension reduction on a value set f_S denoted by $\propto_{-X} f_S$ results in another value set g_T where $T = \propto_{-X} S$, $X \subset dim(S)$, and for every $t \in T$, let $S_t = \{s : \forall v \in dim(T), s(v) = t(v)\}$. Then $g_t = \sum_{S_t} f_s$. A dimension expansion on a value set f_S results in another value set h_Q where $Q = \propto_{+Y} S$ and and for every $s \in S$, let $Q_s = \{q : \forall v \in dim(S), q(v) = s(v)\}$. Then for every $q \in Q_s$, $h_q = f_s$.

Definition 9. Sum/Difference of two value sets is defined as, $f_{S_1} \overset{+}{-} g_{S_2} = h_S$ where $S = S_1 \cup S_2$ and $\forall s \in S$, $h_s = f_s^{'} \overset{+}{-} g_s^{'}$ where $f_s^{'} = f_s$ if $s \in S_1$, 0 otherwise and $g_s^{'} = g_s$ if $s \in S_2$, 0 otherwise. Extension to any number of value sets is trivial.

Definition 10. Product of two value sets is defined as, $f_{S_1} \times g_{S_2} = h_S$ where $S = S_1 \times S_2$ and $\forall s \in S$, $h_s = f_s^{'} \times g_s^{'}$ where $f_s^{'} = f_s$ if $s \in S_1$, 1 otherwise and $g_s^{'} = g_s$ if $s \in S_2$, 1 otherwise. Extension to any number of value sets is trivial.

Definition 11. Join of two value sets is defined as, $f_{S_1} \bowtie g_{S_2} = h_S$ where $S = S_1 \bowtie S_2$ and $\forall s \in S$, $h_s = f_s \times g_s$. Equivalently, $f_{S_1} \bowtie g_{S_2} = \hat{f}_S \times \hat{g}_S$ where $S = S_1 \bowtie S_2$. Extension to any number of value sets is trivial.

Lemma 1. Sum, product, and join operations for value sets are all associative and commutative.

Proof. We will prove that join is associative and commutative. Proofs for other operations can be accomplished in similar ways. First we prove that join is commutative. Let $h_S = f_{S_1} \bowtie g_{S_2}$ and let $s = (s_1, s_2)$ be an arbitrary point in S where s_1 and s_2 are corresponding points in S_1 and S_2 respectively. By definition 5 we know that $\forall v$ if $\exists s_1(v)$ and $\exists s_2(v)$, $s_1(v) = s_2(v)$. Now

considering $h'_{S'} = g_{S_2} \bowtie f_{S_1}$, the point $s' = (s_2, s_1)$ holds the property that $\forall v$ if $\exists s_1(v)$ and $\exists s_2(v)$, $s_1(v) = s_2(v)$. So $s' \in S'$. Furthermore $\forall v(s_1, s_2)(v) = (s_2, s_1)(v)$ and hence $s = s'$ and consequently $s \in S'$. Now with regard to values since $s \in S$ we have $h_s = f_{s_1} \times g_{s_2}$, while $h'_{s'} = g_{s_2} \times f_{s_1} = f_{s_1} \times g_{s_2} = h_s$. This proves that if an arbitrary point and its value are in $f_{S_1} \bowtie g_{S_2}$ then they are in $g_{S_2} \bowtie f_{S_1}$. The other side can be similarly proved and hence $f_{S_1} \bowtie g_{S_2} = g_{S_2} \bowtie f_{S_1}$.

To prove that join is associative let $h_S = (e_{S_1} \bowtie f_{S_2}) \bowtie g_{S_3}$ and $h'_{S'} = e_{S_1} \bowtie (f_{S_2} \bowtie g_{S_3})$. For an arbitrary point $s \in S$, $s = (s_1, s_2, s_3)$ with s_1, s_2 s_3 as corresponding points in S_1, S_2 and S_3 respectively we have,

$$
\begin{array}{lll}
\forall v & \text{if } \exists s_1(v) \text{ and } \exists s_2(v) & \text{then } s_1(v) = s_2(v) \\
 & \text{if } \exists s_1(v) \text{ and } \exists s_3(v) & \text{then } s_1(v) = s_3(v) \\
 & \text{if } \exists s_2(v) \text{ and } \exists s_3(v) & \text{then } s_2(v) = s_3(v) \\
 & \text{if } \exists s_1(v) \text{ and } \exists s_2(v) \text{ and } \exists s_3(v) & \text{then } s_1(v) = s_2(v) = s_3(v)
\end{array}
$$

The above properties also hold for the point $s' = (s_2, s_3, s_1)$ and hence $s' \in S'$ and furthermore $\forall v(s_2, s_3, s_1) = (s_1, s_2, s_3)$ which means $s = s'$ and consequently $s \in S'$. Value-wise we know that $h_s = (e_{s_1} \times f_{s_2}) \times g_{s_3} = e_{s_1} \times (f_{s_2} \times g_{s_3}) = h'_{s'}$ This proves that if an arbitrary point and its value are in $(e_{S_1} \bowtie f_{S_2}) \bowtie g_{S_3}$ then they are in $e_{S_1} \bowtie (f_{S_2} \bowtie g_{S_3})$. The other side can be similarly proved and hence $(e_{S_1} \bowtie f_{S_2}) \bowtie g_{S_3} = e_{S_1} \bowtie (f_{S_2} \bowtie g_{S_3})$.

Lemma 2. Join and product both commute with sum.

Proof. We must show that $h_S = e_{S_1} \bowtie (f_{S_2} + g_{S_3})$ and $h'_{S'} = (e_{S_1} \bowtie f_{S_2}) + (e_{S_1} \bowtie g_{S_3})$ are the same. To do so we first show that $S = S_1 \bowtie (S_2 \cup S_3)$ and $S' = (S_1 \bowtie S_2) \cup (S_1 \bowtie S_3)$ are equal.

First we consider an arbitrary point $s \in S$, $s = (s_1, s_2, s_3)$ with s_1, s_2 s_3 as corresponding points in S_1, S_2 and S_3 respectively. Let $X = dim(S_1), Y =$

$dim(S_2), Z = dim(S_3)$, and $V = X \cup Y \cup Z$.

$$
\begin{array}{llllll}
\forall v \in V & \text{if } v \in X - (Y \cup Z) & & \text{then } s(v) & = & s_1(v) \\
& \text{if } v \in (Y \cup Z) - X & \text{then if } v \in Y - Z & \text{then } s(v) & = & s_2(v) \\
& & \text{then if } v \in Z - Y & \text{then } s(v) & = & s_3(v) \\
& & \text{then if } v \in Z \cap Y & \text{then } s(v) & = & s_2(v) = s_3(v) \\
& \text{if } v \in X \cap (Y \cup Z) & \text{then if } v \in Y - Z & \text{then } s(v) & = & s_1(v) = s_2(v) \\
& & \text{then if } v \in Z - Y & \text{then } s(v) & = & s_1(v) = s_3(v) \\
& & \text{then if } v \in Z \cap Y & \text{then } s(v) & = & s_1(v) = s_2(v) \\
& & & & = & s_3(v)
\end{array}
$$

Now consider the two points $s' = (s_1, s_2)$ and $s" = (s_1, s_3)$ which both belong to S'. Using the similar properties of s' and $s"$ with regard to disjoint sets in the above cases, it is true that $\forall v \in V$, either $s = s'$, or $s = s"$, or $s = s' = s"$. Hence $s \in S'$. Value-wise $e_{s_1} \times (f_{s_2} + g_{s_3}) = (e_{s_1} \times f_{s_2}) + (e_{s_1} \times g_{s_3})$. This proves that if an arbitrary point and its value are in $e_{S_1} \bowtie (f_{S_2} + g_{S_3})$ then they are in $(e_{S_1} \bowtie f_{S_2}) + (e_{S_1} \bowtie g_{S_3})$. The other side can be similarly proved and hence $e_{S_1} \bowtie (f_{S_2} + g_{S_3}) = (e_{S_1} \bowtie f_{S_2}) + (e_{S_1} \bowtie g_{S_3})$.

Lemma 3. A cascade of dimension reductions can be combined into one, and a combined dimension reduction can be separated to a cascade of dimension reductions. This is stated formally as, $\propto_{-V_1} \propto_{-V_2} \ldots \propto_{-V_n} f_S = \propto_{-V} f_S$ where $V = \cup_{i=1}^{n} V_i$, $V \subset dim(S)$, and $\cap_{i=1}^{n} V_i = \emptyset$.

Proof. We show the equivalence in case of two dimension reductions. Extension to higher numbers of dimension reductions is trivial. We want to show that $\propto_{-V_1} \propto_{-V_2} f_S = \propto_{-(V_1 \cup V_2)} f_S$. For the combined dimension reduction we have,

$\propto_{-(V_1 \cup V_2)} f_S = g_T, T = \propto_{-(V_1 \cup V_2)} S$,

and $\forall t \in T, S_t^{1,2} = \{s : \forall v \in dim(T), s(v) = t(v)\}$ and $g_t = \sum_{S_t^{1,2}} f_s$.

Since $dim(T) = dim(S) - (V_1 \cup V_2)$, the above can be rewritten as,

$\propto_{-(V_1 \cup V_2)} f_S = g_T$, $T = \propto_{-(V_1 \cup V_2)} S$ and $\forall t \in T, S_t^{1,2} = \{s : \forall v \in dim(S) - (V_1 \cup V_2), s(v) = t(v)\}$ and $g_t = \sum_{S_t^{1,2}} f_s$.

The first of the cascading dimension reductions can be written as,

$\propto_{-V_2} f_S = g'_{T'}$, $T' = \propto_{-V_2} S$ and $\forall t' \in T', S_{t'}^2 = \{s : \forall v \in dim(S) - V_2, s(v) = t'(v)\}$ and $g'_{t'} = \sum_{S_{t'}^2} f_s$.

Cascading with the second dimension reduction gives,

$\propto_{-V_1} (\propto_{-V_2} f_S) = g"_{T"}$, $T" = \propto_{-V_1} T'$ and $\forall t" \in T", S_{t"}^1 = \{s : \forall v \in dim(T') - V_1, s(v) = t"(v)\}$ and $g_{t"} = \sum_{S_{t"}^1} g'_s$.

We can write $T" = \propto_{-V_1} T' = \propto_{-V_1 - V_2} S = \propto_{-(V_1 \cup V_2)} S$. Also $g'_s = \sum_{S_{t'}^2} f_s$. So the above relation can be rewritten as,

$\propto_{-V_1} (\propto_{-V_2} f_S) = g"_{T"}$, $T" = \propto_{-(V_1 \cup V_2)} S$ and $\forall t" \in T", S_{t"}^1 = \{s : \forall v \in dim(T') - V_1, s(v) = t"(v)\}$ and $g_{t"} = \sum_{S_{t"}^1} \sum_{S_{t'}^2} f_s$.

Now given that $\sum_{S_{t"}^1} \sum_{S_{t'}^2} f_s = \sum_{S_t^{1,2}} f_s$ it follows that $g_t = g"_{t"}$ and our equivalence is proved.

Lemma 4. Commuting dimension reduction with join. Let f_{S1} and g_{S2} be two value sets. Let $\propto_{-(X1 \cup X2)}$ be a dimension reduction where $X1 \subset dim(S1)$, $X2 \subset dim(S2)$, $X1 \cap X2 = \emptyset$, $X1 \cap dim(S2) = \emptyset$, and $X2 \cap dim(S1) = \emptyset$. Then

$$\propto_{-(X1 \cup X2)} f_{S1} \bowtie g_{S2} = \propto_{-X1} (f_{S1}) \bowtie \propto_{-X2} (g_{S2})$$

Proof. We suffice to outline the stepwise proof as follows.

Step 1: Assume that $f_{S1} \bowtie g_{S2} = e(S)$.

Step 2: Show that

$$f_{S1} \bowtie \propto_{-X2} g_{S2} = \propto_{-X2} e(S).$$

Note that the join equality constraints that apply to the intersection of dimensions of $S1$ and $S2$ are not affected by such a dimension reduction because $X2$ does not intersect with $S1 \cap S2$.

Step 3: Using the same reasoning presented in step 2 show that

$$\propto_{-X1} f_{S1} \bowtie \propto_{-X2} g_{S2} = \propto_{-X1} (\propto_{-X2} e(S))$$

Step 4: Using Lemma 3 (cascading dimension reductions) we conclude that

$$\propto_{-X1} f_{S1} \bowtie \propto_{-X2} g_{S2} = \propto_{-(X1 \cup X2)} e(S) = \propto_{-(X1 \cup X2)} (f_{S1} \bowtie g_{S2})$$

3.2 Value Sets Join Chain Optimization Problem

Using the above operations we can form value sets algebraic expressions. A typical expression is a sum of products of value sets f_S's which is denoted by $e = \sum \pi f_S$. We will later show that probabilistic inference problems can be described as the operation of dimension reduction on joins of such sums of products.

An optimal way of performing such operations finds the resultant value set at the minimum cost. We assume that sum and dimension reduction operations on value sets have no cost. For product or join operations the cost is the number of real number multiplications involved in performing those operations. Formally the problem is defined as follows.

Definition Given

1. A set of m variables V,
2. A set of n subsets of V : $R = \{R_{\{1\}}, R_{\{2\}}, \ldots, R_{\{n\}}\}$,
3. For every $R_i \in R$, a finite collection of sets $F^i_{S_i} = \{F^{i,1}_{S_{i,1}}, F^{i,2}_{S_{i,2}}, \ldots F^{i,J}_{Si,J}\}$, where every element is a set of value sets $F^{i,j}_{S_{i,j}} = \{f^{i,j,1}_{S_{i,j,1}}, f^{i,j,2}_{S_{i,j,2}}, \ldots f^{i,j,K}_{Si,j,K}\}$ defined over the super domain of R_i, and a sum of products of value sets in $F^{i,j}_{S_{i,j}}$:

$$e_i = \sum_{j=1}^{j \leq J} \prod_{k=1}^{k \leq K} f^{i,j,k}_{S_{i,j,k}}$$

Define operation:

1. A sequence of ℓ joins over $\ell + 1$ value sets with dimension reduction,

$$\beta_\ell = \propto_{V_I} \bowtie_{i=1}^{\ell+1} e_i, V_I \subset (\cup_{i=1}^{\ell+1} V_i)$$

2. The cost function of β_ℓ given the operations order σ is denoted by $\mu_\sigma(\beta_\ell)$ and it is defined as the number of multiplications necessary to obtain the resultant value set if the operations are performed in the order specified by σ.

The value sets join chain optimization problem is to find σ such that $\mu_\sigma(\beta_n)$ is minimal.

3.3 Inference in Bayesian Networks and Value Sets Algebra

Probabilistic inference in Bayesian network can be described as the problem of marginalizing a joint distribution of the involved variables. For example, in the Bayesian network shown in figure 1 we can describe the probability of A as,

$$P(A) = \sum_{BD} P(ABD) = \sum_{BD} P(D/AB)P(A)P(B)$$

Lets assume that all the variables have the binary domain $\{t, f\}$. Furthermore we assume that D has a noisy-or interaction with A and B. This leads to factorization of $P(D/AB)$ as indicated earlier.

$$P(D/AB) \quad = \quad 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}}$$

The calculation of $P(A)$ can be described in terms of value sets algebra in the following way. We describe every probabilistic distribution as a value set whose domain point set is the state-space of the distribution, a subset of the super domain of the set of involved variables. The range of such value sets is [0,1] subset of real numbers with the actual mappings being the probability values indicated in the probability distribution. For example 1_{D_t} is a value set whose point set is $\{D = t\}$ and the value mapped to the single point is 1. Also $c'_{D_t|A_{t,f}}$ is a value set whose point set is $\{(D = t, A = t), (D = t, A = f)\}$ with the values $1 - c_A(D)$ for $(D = t, A = t)$ and 1 for $(D = t, A = f)$. If we denote by $V(X/Y)$ the value set corresponding to the distribution $P(X/Y)$ then we can describe the probability calculations for $P(A)$ in terms of value set algebra as,

$$\propto_{-BD} V(D/AB) \bowtie V(A) \bowtie V(B)$$

which is equivalent to

$$\propto_{-BD} (V(D_t) - V(D_t/A) * V(D_t/B) + V(D_f/A) * V(D_f/B)) \bowtie V(A) \bowtie V(B)$$

Performing the value set operations will result in a value set whose domain point set is $\{At, Af\}$ and its mappings are the corresponding probabilities.

Chapter 4

COHERENCY OF LOCAL EXPRESSIONS

The various properties of a noisy-or or an asymmetric interaction make it possible to represent the joint conditional distribution (JCD) in a very compact form in a local expression. To be coherent, the entries on every row in a JCD must add up to 1. So, an obvious way to check the coherency of a local expression is to evaluate the expression using the rule in Chapter One to obtain the JCD table and numerically check every row. But this is expensive since the number of rows is exponential in the number of parents.

A better approach to coherency checking is to take advantage of the decompositions and factorings that are present in the expression to reduce the evaluations and checkings of the actual numbers. To simplify our coherency checking operations, we require that expressions be represented in a canonical form, as defined below.

4.1 Canonical Form

Canonical forms have been employed in computer algebra systems to simplify certain kinds of operations on polynomials. We define a canonical form for local expressions in the following way to facilitate the coherency checking on such expressions.

1. An expression in canonical form is the sum of some terms and every such term is the product of some factors.

2. If the conditioned variable has a full scope in the state-space of a factor then the factor must be probabilistically coherent.

Note that in the contexts where the symbolic manipulation of local expressions is discussed we mostly use the word "factor" to refer to "distribution" in the local expression grammar. The same entity may be represented and referred to as "value set" in more theoretic discussions.

The second condition can be verified on a factor by reducing the dimension of the corresponding value set. In this reduction the scope of the conditioned variable in the dimension will be reduced to the empty set. Then the condition is verified by observing that all resultant values are one. There are local expressions that are numerically coherent but are not in canonical form. One example of such expressions is the expression for base and modifier representation of additive value function (D'Ambrosio, 1995). Using the weighted sum approach of (Dagum & Galper, 1993) for the additive relationships will give a local expression that is in canonical form.

4.2 Coherency Verification Procedure

Given an expression whose coherency is in question, the following procedure is performed. Basic conjunctive state-space (BCS) is formed for every term. A term BCS is the joint state-space of the factors in the term considering only the variables that are explicitly present in the dimension of the term factors. If the conditioned variable's scope in BCS is full, and BCS is disjoint from other terms BCS, then the term is a prime term. Prime terms are individually verified to be coherent. For example, consider the following local expression representing part of an expression for a finding node F with two parents $D1$

and $D2$. It is assumed that relationship among the nodes is of a noisy-max (Henrion, 1988); (Pradhan, Provan, Middleton, & Henrion, 1994), and all nodes have the ternary domain of low, medium, and high.

$$\begin{aligned} exp(f) &= 1_{F_{l,m,h}|D2_l} * f1_{F_{l,m,h}|D1_{l,m,h}} + 1_{F_{l,m,h}|D1_{m,h}} * f2_{F_{l,m,h}|D2_l} \\ &\quad + f'1_{F_{l,m}|D1_{m,h}} * f'2_{F_{l,m}|D2_{m,h}} \end{aligned}$$

There are three terms in this expression. The first term BCS, using the local expressions subscript notation, is $F_{l,m,h}D2_lD1_{l,m,h}$. The BCS for the second term is $F_{l,m,h}D1_{m,h}D2_l$. Both terms are prime terms, and each is individually verified to be coherent.

For all the other terms where the conditioned variable has only a partial scope in BCS, the following operations are performed. First, if the size of the conditioned variable's scope is greater than 1 in any term, the term will be rewritten. Such a term will be the sum of some terms, one for each distinct element of the conditioned variable domain. In our example the factors of the last term will be rewritten that way to get an equivalent expression,

$$(f1_{F_l|D1_{m,h}} + f"1_{F_m|D1_{m,h}}) * (f2_{F_l|D2_{m,h}} + f"2_{F_m|D2_{m,h}})$$

which is converted into canonical form to give,

$$f1_{F_l|D1_{m,h}} * f2_{F_l|D2_{m,h}} + f"1_{F_m|D1_{m,h}} * f"2_{F_m|D2_{m,h}}$$

Through dimension reduction the conditioned variable is removed from the dimensions of all the terms. The terms are then grouped into sets which are disjoint in terms of state-space. Hence terms within any such group or set share the same state-space. Continuing with the example we will have,

$$f1_{D1_{m,h}} * f2_{D2_{m,h}} + f"1_{D1_{m,h}} * f"2_{D2_{m,h}}$$

Both remaining terms have the same state-space of $D1_{m,h}D2_{m,h}$. The only remaining operation in our current example is to perform the products and sum all the terms against one in the state-space. In general within any such set with shared state-space a search will take place to find and match pairs of terms with more than one factor and opposite signs. These are matched factor by factor for symbolic reductions. For example consider the noisy-or expression of Section 2:

$$e(D) = 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}}$$

After this expression goes through the stages of our procedure that are already described, it will be as follows:

$$1 - c'_{A_{t,f}} * c'_{B_{t,f}} + c'_{A_{t,f}} * c'_{B_{t,f}}$$

Note that the second and third term share state-space. The equality of factors is established based on equality of their state-space and then the corresponding values. If such a one-to-one factor match is established the terms will be removed. This will reduce the above example expression to 1. Generally, and in case there are other remaining terms, for the remaining terms in the group the products will be performed in an order found through a factoring algorithm. Then all terms are summed up against one in all the points in the state-space.

All the transformations prescribed by the coherency verification procedure are correct in the sense that their application to any expression maintains the coherency of the original expression.

We must also mention that the need to efficiently multiply distributions shows that in the worst case the coherency verification can be as hard as

general probabilistic inference. The real sources of efficiency in both problems are the same. First there are structural factorizations and decompositions over the set of all variables that follow from true probabilistic independencies among them. Second, there are algebraic properties in the distributions at the local levels of variables interactions. They both allow one to order the operations in a cost efficient manner usually by keeping the size of the intermediate results at minimum. It is a plausible hypothesis to state that both coherency verification and probabilistic inference share the efficiency benefits of the above sources. Therefore, the easier coherency verification on a local expression is, the easier probabilistic inference with that local expression is, and vice versa.

Chapter 5

LOCAL EXPRESSIONS INFERENCE

The problem of inference with extended local expressions is characterized as a combinatorial optimization problem. An old solution to a restricted version of the problem is analyzed as a heuristic search and its essential features are enhanced and modified to form a heuristic solution for our new problem.

5.1 Introduction

Local expressions represent the various forms of local interactions and dependencies among adjacent nodes in Bayesian networks. More specifically, a local expression in a node represents the joint conditional probability distribution (JCD) of the node variable given its parents. Local expression languages are expressive and flexible, so that local expressions can represent more dependency structures including noisy OR and asymmetric dependencies. Such dependencies can not be easily represented in standard Bayesian network graph or distributions. Local expressions enhance Bayesian networks with these representational capabilities and demand an enhanced inference mechanism that can take advantage of new finer grain relationships.

ESPI that was introduced in Chapter One was an earlier attempt (D'Ambrosio, 1995) at this problem that used different heuristics and strategies than what we will represent here. Some other approaches have been proposed to deal with finer grain relationships in Bayesian networks as well. For example in the context of Intercausal Independencies (ICI) there are two major

approaches. Heterogeneous Factorization (Zhang & Poole, 1996); (Zhang & Poole, 1994) and temporal Belief Networks (Heckerman & Breese, 1994) are the major approaches which are capable of representing many forms of causal independencies. Many theoretical and practical modeling aspects of ICI are covered in (Pearl, 1988), (Pradhan et al., 1994), and (Srinivas, 1993). There are also approaches that deal with asymmetries and contingencies in Bayesian network and influence diagram inference. See (Geiger D. & Heckerman D., 1991), (Fung R., & Shachter R., 1991), (Smith, Holtzman, & Matheson, 1993), (Shenoy, 1996), and (Boutilier et al., 1996). Many of these approaches try to eliminate the artificial and unnecessary computations which follow from ad-hoc modeling of asymmetries in standard Bayesian networks.

This thesis responds to the challenge from a combinatorial optimization perspective. Earlier, a formal characterization of this problem was presented in Chapter Three. In this chapter We will present an informal but nonetheless intuitive description of the problem as an optimal search problem. Then we review Optimal Factoring Problem that is closely related to our current problem under the same light. Finally using our search analogy we present a reasonable heuristic solution for the problem.

5.2 Local Expressions Inference as an Optimal Search Problem

The inputs to the algorithm are a Bayesian network graph, a set of local expressions representing the distributions in that network, and a query. The algorithm goal is to combine the expressions and find the response to the query with the minimum number of multiplications. We assume that the network is always refined to a subnetwork consisting of only those nodes that are relevant to the query. Using d-separation, such a refinement can be done in a time

polynomial in the number of nodes (Geiger, D., Verma, T., & Pearl, J., 1989). We begin with an intuitive, albeit informal description of the problem as a combinatorial optimization problem. Then we will review the Optimal Factoring which is a closely related problem. We will discuss how we can extend and apply the ideas from the heuristic solution of optimal factoring to our current problem.

Assume that we can apply two operations to local expressions: *combination* and *reduction*. Combining two local expressions such as e_i and e_j results in an expression e_{i*j} such that if

$$e_i = t_{i,1} + t_{i,2} + \ldots + t_{i,n}$$

$$e_j = t_{j,1} + t_{j,2} + \ldots + t_{j,m}$$

Then,

$$e_{i*j} = \sum_{k=1,2,\ldots,n}^{l=1,2,\ldots,m} t_{k,l}$$

$$t_{k,l} = t_{i,k} * t_{j,l}$$

$$\forall i, j \rightarrow i = 1, 2, \ldots, n; j = 1, 2, \ldots, m$$

The state-space of a result term such as $t_{k,l}$ is constrained to the intersection of the state-spaces of the operand terms $t_{i,k}$ and $t_{j,l}$. For example in the network

in Fig. 2.1 we had the following local expressions.

$$\begin{aligned} e(D) &= 1_{D_t} - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} \\ &\quad + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}} \\ e(A) &= A_t + A_f \\ e(B) &= B_t + B_f \end{aligned}$$

Combining expressions e(A) and e(D) will give,

$$\begin{aligned} e(A * D) &= 1_{D_t} * A_t - c'_{D_t|A_t} * c'_{D_t|B_{t,f}} * A_t \\ &\quad + c'_{D_f|A_t} * c'_{D_f|B_{t,f}} * A_t \\ &\quad 1_{D_t} * A_f - c'_{D_t|A_f} * c'_{D_t|B_{t,f}} * A_f \\ &\quad + c'_{D_f|A_f} * c'_{D_f|B_{t,f}} * A_f \end{aligned}$$

Note that in the first half of terms in the result all instances of A's dimension are constrained to t. The terms in the second half of the expression include only f as a dimension for A.

The combination operation does not include any multiplication or summation. Reduction, on the other hand, includes multiplication of the factors in terms and summation of terms. A complete reduction will perform all multiplications and summations indicated in expression. But the algorithm may decide to apply a partial reduction to an expression or be lazy and apply no reduction at all. A partial reduction means only some of multiplications and summations in a given expression are performed. In the network in Fig. 2.1, suppose the given query involves all the variables and reductions are limited to either complete reductions or no reductions. Then the initial parts of the search space for finding a minimal cost solution are depicted in Fig. 5.1.

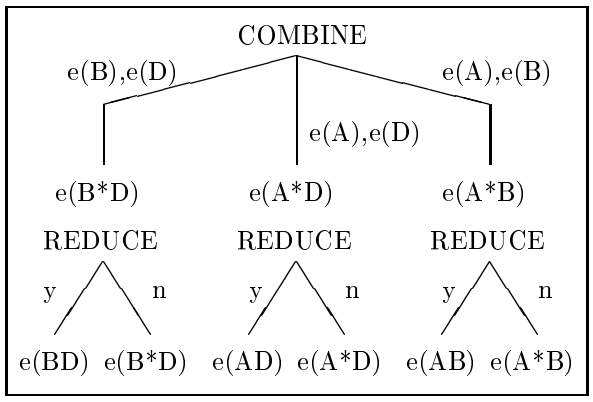

FIGURE 5.1: Part of the search space for finding the optimal order of operations

We can consider combination and reduction as one operation. In that case the number of branches for a node will depend on the number of node pairs to combine as well as the number of possible reductions on each pair.

5.3 A Review of Optimal Factoring Problem and its Heuristic Solution

Optimal Factoring Problem (OFP)(Li & D'Ambrosio, 1994) is how to combine distributions in a standard Bayesian network into a query response at the minimum cost. OFP is a restricted form of optimal inference with local expressions. If local expressions are restricted to include only one term which in turn has only one factor then optimal inference with local expressions will be the same as OFP. OFP has a heuristic algorithm that performs well. We will revisit it in order to come up with good heuristics for our problem. The heuristic solution to this problem treats the distributions as subsets of network variables. For example, $p_{E|A,B}$ will be represented as $\{A, B, E\}$. These subsets are also referred to as factors. Here is a description of the OFP heuristic algorithm.

1. Divide the set of factors into independent (i.e., no shared variables) subsets if possible. Solve each subset separately using the following method, then combine the results:

2. Form a *factor set* A which includes every factor that may be chosen for the next combination (initially all the relevant net distributions). Each factor in set A is represented as a set of variables. Initialize a *candidate set* B empty.

3. Add all pairwise combinations of factors of the factor set A to B, provided that these combinations are not already in set B and additionally one factor of the pair contains a variable which is a parent or child of a variable in the second factor. Compute $u = (x \cup y)$ and $sum(u)$ of each pair, where x and y are factors in the set A, $sum(u)$ is the number of variables in u which can be summed out when the product corresponding to combination of the two factors is performed. A variable can be summed out when it appears in neither the set of target variables nor any of the factors not in the current pair.

4. Choose elements from set B such that $C = \{u|u : minimum_B(|u| - sum(u))\}$, here $|u|$ is the size of u excluding observed variables. If $|C| = 1$, x and y are the factors for the next combination; otherwise, choose elements from C such that $D = \{u|u : maximum_C(|x| + |y|),\ x, y \in u\}$. If $|D| = 1$, x and y are the factors for the next multiplication, otherwise, choose any one of D.

5. Generate a new factor by combining the pair chosen in the above steps. Modify the factor set A by deleting the two factors in the chosen pair from the factor set and adding the new factor in the set.

6. Delete any pair in set B which has non-empty intersection with the candidate pair.

7. Repeat step 3 to 6 until only one element is left in the factor set A which is the final result.

The algorithm repeatedly selects the best related pair of factors to combine until all factors are combined. The best pair is identified as the one with the least remaining size. If more than one candidate pair have the minimum remaining size then the one with the largest original size will be selected.

In our view the OFP heuristic is a restricted instance of A* search in the space of all possible solutions. The algorithm trades off the current cost function $g(n)$ with efficiency by ignoring it and becomes a greedy algorithm. For the future cost function $h(n)$ the algorithm uses the remaining size (and the sum of original sizes to break ties) heuristic. Since the cost of combining two distributions is the product of their corresponding sizes it is trivial to show that remaining size is an admissible heuristic. In case of ties using the combination of larger partial solutions guarantees that the remaining parts of the solution will be smaller. These smaller parts are going to be combined with the current solution that has the least remaining size and this makes the criteria admissible. We must also bear in mind that OFP ignores $g(n)$ and hence is not guaranteed to find the optimal cost solution as a search algorithm.

5.4 SPI*: A Heuristic Algorithm for Local Expressions Inference

In comparison with OFP we are dealing with sums of products of factors in every expression as our units to be combined. So to be able to use a similar heuristic we have to provide the following items: (1) A scoring criterion for combination candidates. (2) A set of decision making criteria as to carry out any products in term-by-term combinations or not at any time. (3) A scheme to indicate when to sum out the variables which are not the target of the query.

ESPI, an earlier (D'Ambrosio, 1995) attempt at this problem used different heuristics and strategies than what we will represent here. In that approach heuristic scoring function for candidate selection was extended to include number of factors in expressions. Our approach is to revert to the base function and use the actual size of all factors in an expression. Also with regard to combination versus reduction operations the earlier attempt was to use an immediate all out reduction strategy. Meaning that right after any two expressions are combined all the products in resultant terms are carried out. Our strategy is to postpone the products until all the relevant factors are present for the best product ordering unless performing them now benefits the overall cost in a sensible way. We will refer to our new heuristic as SPI* algorithm.

5.4.1 Adapting the Remaining Size Criterion

We would like to use the same heuristic cost functions that were used in OFP. Every local expression in OFP, however, was limited to one single-factor term, and that is not the case in extended local expressions that we have to deal with. So we need to define what we mean by "remaining size" for the extended local expressions. The remaining size for an extended local expression is defined as

the total size of all individual factors that remain in all terms of the expression. It must be obvious that this definition favors factored representation of distributions against their non-factored representation.

In general the remaining size depends on the products or sums in the expression that may be performed after combination of expressions. Some of such operations are safe to perform in the sense that postponing them certainly will not reduce the total cost. We will identify such operations later so that an upper bound on the remaining size for a combination can be found. Yet some other operations can not be ruled safe to perform, even though to perform them in conjunction with summation of terms may reduce the remaining size for the current expression. In other words, postponing such operations may serve the total cost reduction better than performing them to reduce the remaining size. We will introduce a heuristic way for making such decisions in SPI* algorithm.

5.4.2 Performing the Operations in Expressions

Combination operation first forms a term-by-term symbolic product of the operand expressions as described in Section 5.2.1. As we mentioned, no actual product or sum is performed as part of the combination. Then heuristically reasonable operations among the factors and terms in the product are performed. There are three types of operations that are heuristically reasonable and they will be described here.

5.4.2.1 Marginalization Related (MGR) Operations

The first type of operations that are considered reasonable to perform are those operations among the factors in a term that are necessary to form a joint factor,

so that some variables can be safely summed out. The following definitions help identify what variables can be safely summed out of terms.

F is the set of factors in a term. C is a subset of F which contains all the conditional factors in F. J is the set of conditional factors in F such that if all factors in F are multiplied, nothing will remain conditioned on the conditioning variables of J factors. In other words, J is a subset of C such that all conditioning variables of factors in J are present in the factors of $F - C$. V_g is the set of variables in the factor set G. VI_g is defined as a subset of V_g where elements have no link to variables outside V_g. S_g is a subset of V_g whose elements are to be summed out in a query. SI_g is a subset of S_g where elements have no link to variables outside V_g.

A term t_1 can be marginalized with regard to variables in its SI_j. Restricting marginalizations to the variables in SI_j guarantees that all the factors relevant to such variables are present in the term. It also guarantees that the most efficient way of multiplying these factors now at the first opportunity is indeed the most efficient way of multiplying them anywhere during the rest of the inference computations.

In order to perform the marginalization for every variable si in SI_j we find the subset of J denoted by J_{si} which includes factors involving si. Then for every factor in J_{si} such as f_j we find the corresponding factor in $F - C$ or f_u which can be multiplied by f_u to give an unconditional factor f_{uj}. Now f_{uj} can be marginalized with regard to si and si can be summed out of f_{uj}.

Our heuristic marginalizes more aggressively. In any term every variable that is neither a target variable in the query nor present in other expressions can be marginalized. In a term all the factors involving such variables are joined and the joint factor is marginalized with regard to those variables.

5.4.2.2 Subset-Dimension Related (SDR) Operations

The second type of operations that are considered reasonable are the binary operations in which one factor's dimension is a subset of other factor's dimension. So if such an operation is performed, it produces a factor whose dimension is the same as the dimension of one of the operands. For example, $A * F_{t,f}E_t/A$ is a SDR operation. But if there is a factor such as D_t/A in a related expression, then it would be better to wait until the product $A * D_t/A$ is performed first. Heuristically, we consider that if the product $f1 * f2$ is an SDR operation, then we need to justify it against every factor $f3$ in the fringe of the current expression that intersect with either $f1$ or $f2$. So $f1$, $f2$, and $f3$ form a triangle where all possible orders among the products are tried to see if SDR operation is justified or not.

5.4.2.3 Term Summation Related (TSR) Operations

There are times that joint state-spaces of several terms in an expression are the same. In other words, if all the products in those terms are performed, the resultant factors will have the same state-spaces. Such factors can be summed out, resulting in a smaller but equivalent expression. On the other hand, performing all the products in such terms may eliminate some opportunities for cheaper combinations of the factors in those terms and the factors in other expressions. So a decision has to be made whether to perform all products in same state-space terms and sum them up or postpone such products until the expression is combined with other expressions.

Let us restrict this problem to a situation where we can consider that the impact of the decision to sum or not to sum terms can be best evaluated based on the situation in the neighborhood of the expression rather than the whole

network. Further, assume that there is only one group of the same state-space terms in our expression and this group is denoted by $t_1^+, t_2^+, \ldots, t_n^+$. We will denote this set as et^+. The neighborhood of an expression is the set of expressions which contain at least one variable which is a parent or child of a variable in the expression. Let $t_1^{sdr}, t_2^{sdr}, \ldots, t_m^{sdr}$ denote such neighborhood expressions where their terms are reduced to a subset of their factors. Every remaining factor in those terms has a set of variables that is a subset of the set of variables in at least one factor of any t_i^+. We will denote the set of t^{sdr}'s by et^{sdr}. The set et^{sdr} represents the expressions whose terms include factors whose combinations with factors in any t_i^+ do not result in an increased dimension or larger state-space.

If et^{sdr} is empty, then all the products necessary for the summation of terms in et^+ will be performed and they will be summed up. Otherwise, we consider et^{sdr} as a set of expressions and find the following two costs. Cost-1 is the cost of performing all products in et^+ and then combining the resultant expression with et^{sdr} and performing the remaining products. Cost-2, is the cost of performing all the products after et^{sdr} and et^+ are symbolically combined. If Cost-1 > Cost-2, then products in et^+ should not be performed, otherwise they must.

SPI* heuristic considers that TSR operations are either all done or all postponed in an expression. Furthermore, it restricts the evaluation context to individual expressions in the fringe of the expression. This evaluation is implemented at the candidate generation step by generating all possible summed-terms and unsummed-terms combinations of the candidate pair.

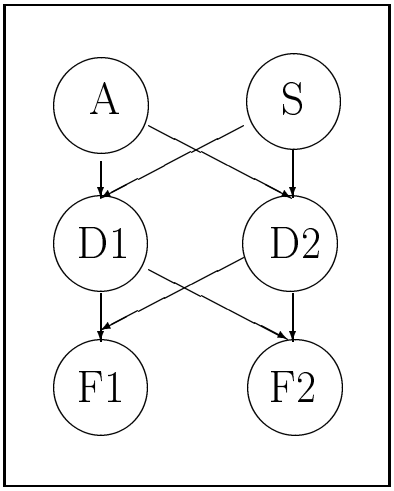

FIGURE 5.2: An example network for the heuristic algorithm.

5.5 An Example Execution of SPI* Algorithm

The execution of the algorithm will be briefly demonstrated using the Bayesian network in Fig. 5.2. In this network variable S stands for the sex of the patient which is either female or male represented by the domain {f, m}. Variable A represents the age group of the patient which can be young, middle-age, or old represented by $\{y, m, o\}$. There are two diseases represented by D1 and D2. Their absence or presence is represented by true or false in the domain $\{t, f\}$. The same domain exists for the two findings F1 and F2 in the network. If sex is female, D2 is false. If age is young, then D1 is false. F1 has a noisy-or relationship with D1 and D2. The individual influences in that relationships are denoted by q11 and q12 for D1 and D2 respectively. The same type of relationship holds between F2 and its parents D1 and D2 with similar influences q21 and q22 for D1 and D2 respectively. Local expressions in the original

network are as follows.

$$
\begin{aligned}
e(A) &= p1_{A_y} + p2_{A_m} + p3_{A_o} \\
e(S) &= p4_{S_m} + p5_{S_f} \\
e(D1) &= 1_{D1_f/A_y} + p6_{D1_{t,f}/S_{m,f}A_{m,o}} \\
e(D2) &= 1_{D2_f/S_f} + p7_{D2_{t,f}/S_m A_{y,m,o}} \\
e(F1) &= 1_{F1_t/D1_{t,f}} * 1_{F1_t/D2_{t,f}} - q11_{F1_t/D1_{t,f}} * q12_{F1_t/D2_{t,f}} \\
&\quad + q11_{F1_f/D1_{t,f}} * q12_{F1_f/D2_{t,f}} \\
e(F2) &= 1_{F2_t/D1_{t,f}} * 1_{F2_t/D2_{t,f}} - q21_{F2_t/D1_{t,f}} * q22_{F2_t/D2_{t,f}} \\
&\quad + q21_{F2_f/D1_{t,f}} * q22_{F2_f/D2_{t,f}}
\end{aligned}
$$

Consider the query: P(D1=t/F1=t,F2=f,A=m,S=m) = ? Local expressions are reduced to:

$$
\begin{aligned}
e(A) &= p2_{A_m} \\
e(S) &= p4_{S_m} \\
e(D1) &= p6_{D1_t/S_m A_m} \\
e(D2) &= p7_{D2_{t,f}/S_m A_m} \\
e(F1) &= 1_{F1_t/D1_t} * 1_{F1_t/D2_{t,f}} - q11_{F1_t/D1_t} * q12_{F1_t/D2_{t,f}} \\
e(F2) &= q21_{F2_f/D1_t} * q22_{F2_f/D2_{t,f}}
\end{aligned}
$$

In the first loop candidates and their scores are as follows.

Candidates	Remaining-Size
e(A), e(D1)	1
e(A), e(D2)	3
e(S), e(D1)	1
e(S), e(D2)	3
e(D1), e(F1)	3
e(D1), e(F2)	3
e(D2), e(F1)	6
e(D2), e(F2)	6

It is assumed here that ties are arbitrarily broken and as a result $e(A), e(D1)$ will be selected to produce the following combination.

$e(A * D1) = p2_{A_m} * p6_{D1_t/S_m A_m} = p8_{D1_t A_m/S_m}$

where $p8 = p6 * p2$. The second loop candidates are as follows.

Candidates	Remaining-Size
e(AD1), e(D2)	2
e(S), e(AD1)	2
e(S), e(D2)	3
e(AD1), e(F1)	3
e(AD1), e(F2)	3
e(D2), e(F1)	6
e(D2), e(F2)	6

As a result $e(ASD1) = p9_{D1_t S_m A_m}$ will be selected where $p9 = p8 * p4$. The next loop candidates are as follows.

Candidates	Remaining-Size
e(ASD1), e(D2)	2
e(ASD1), e(F1)	4
e(ASD1), e(F2)	4
e(D2), e(F1)	6
e(D2), e(F2)	6

As a result $e(ASD1D2)$ will be selected. The next loop candidates are as follows.

Candidates	Remaining-Size
e(ASD1D2), e(F1)	2
e(ASD1D2), e(F2)	2

As a result $e(ASD1D2F1)$ will be selected. A term summation in $e(F1)$ takes place before the combination and the reduction. The final combination is with the remaining $e(F2)$ expression. The total evaluation cost for the obtained ordering in this example is 17 multiplications.

5.6 Performance of the Heuristic

SPI* has been implemented and tested with many networks. In this section the performance of SPI* on several interesting cases will be presented and compared with performances of other relevant approaches.

5.6.1 Asymmetries and Contingencies

The oil wildcatter problem The oil wildcatter problem (Raiffa, 1968) contains asymmetric as well as contingent relationships. An influence diagram

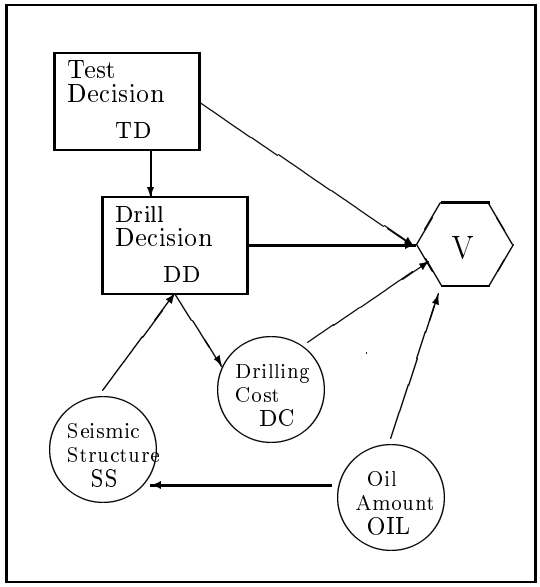

FIGURE 5.3: Influence diagram for the oil wildcatter example.

for the problem is shown in Fig. 5.3. Since the description of the problem is well known, we will suffice to describe it through the following local expressions.

$$
\begin{aligned}
e(TD) &= TD_{yes,no} \\
e(OIL) &= OIL_{dry,wet,soaking} \\
e(SS) &= SS_{no-struct,open,closed}/OIL_{dry,wet,soaking} \\
e(DD) &= DD_{yes,no}/TD_{no}SS_{\emptyset} + DD_{yes,no}/TD_{yes}SS_{no-struct,open,closed} \\
e(DC) &= DC_{low,med,high}/DD_{yes} + DC_{\emptyset}/DD_{no} \\
e(V) &= V_{t,f}/DD_{yes}DC_{low,med,high} + V_{t,f}/DD_{yes}OIL_{dry,wet,soaking} \\
&\quad + V_{t,f}/TD_{yes,no}
\end{aligned}
$$

The natural inference problem for this type of decision problems is to find the optimal policies. On the Bayesian network of this example we must find the

joint probability distribution of all decision nodes and their parents given that value is observed to be true (Shachter, R. D. & Peot, M. A., 1992). A symmetric version of this problem built using artificial states for variables required 200 multiplications for the query to be computed. ESPI, on the other hand, took 151 multiplications when local expressions are used to model the asymmetries and contingencies. SPI* found an evaluation for the query that takes only 96 multiplications.

The Car Buyer problem The car buyer problem (Howard, 1989) is another well-known asymmetric problem. The influence diagram for the problem is shown in Fig. 5.4. The following local expressions suffice to describe the problem here.

$$
\begin{aligned}
e(CC) &= CC_{peach,lemon} \\
e(R1) &= R1_{\emptyset}/T1_{no-test} + R1_{zero,one}/T1_{steer,trans}CC_{peach,lemon} \\
&\quad + R1_{zero,one,two}/T1_{fuel-elect}CC_{peach,lemon} \\
e(R2) &= R2_{\emptyset}/T1_{no-test,steer,fuel-elect} + R2_{\emptyset}/T2_{no} + R2_{\emptyset}/R1_{two} \\
&\quad + R2_{zero,one}/T1_{trans}T2_{yes}R1_{zero,one}CC_{peach,lemon} \\
e(T1) &= T1_{no-test,steer,fuel-elect,trans} \\
e(T2) &= T2_{no}/T1_{no-test,steer,fuel-elect} + T2_{yes,no}/T1_{trans}R1_{zero,one} \\
e(B) &= B_{y,n,g}/T1_{no-test} + B_{y,n,g}/T1_{steer}R1_{zero,one} \\
&\quad + B_{y,n,g}/T1_{fuel-elect}R1_{zero,one,two} \\
&\quad + B_{y,n,g}/T1_{trans}R1_{zero,one}T2_{yes,no}R2_{zero,one} \\
e(V) &= V_{t,f}/T1_{no-test,steer,fuel-elect,trans} + V_{t,f}/T2_{yes,no} \\
&\quad + V_{t,f}/B_{y,n,g}CC_{peach,lemon}
\end{aligned}
$$

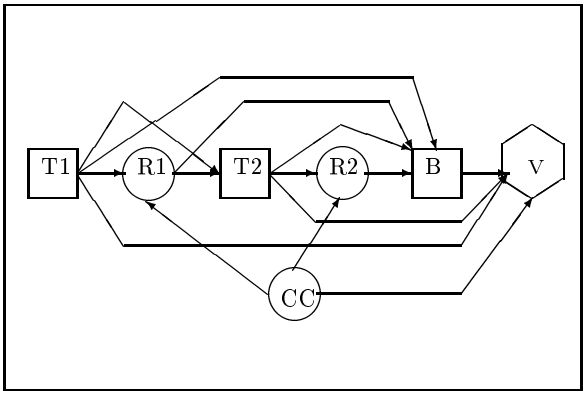

FIGURE 5.4: Influence diagram for the car buyer example.

We computed optimal policies for this example in the same way described earlier. The symmetric version needed 960 multiplications. ESPI took 519, while SPI* presented an evaluation with 329 multiplications.

The Generalized Buying Problem The Generalized Buying Problem was defined in(Qi & Poole, 1995) as follows.

> Suppose we have to decide whether to buy an expensive and complex item. Before making the decision, we can have n tests, denoted by $T_1,\ldots,T_n$, on the item and we can look at the corresponding results denoted by $R_1,\ldots,R_n$. Suppose test T_i has k_i alternatives and j_i possible outcomes for $i = 1,\ldots,n$. Among k_i alternatives for test T_i, one stands for the option of no-testing. Correspondingly, among the j_i possible outcomes of test T_i, one stands for no-observation, resulting from the no-testing alternative.

An influence diagram for this problem is shown in Fig. 5.5. Our objective is to show how effectively our heuristic exploits asymmetry for this class of decision problems. We will compare the cost of computing optimal policies

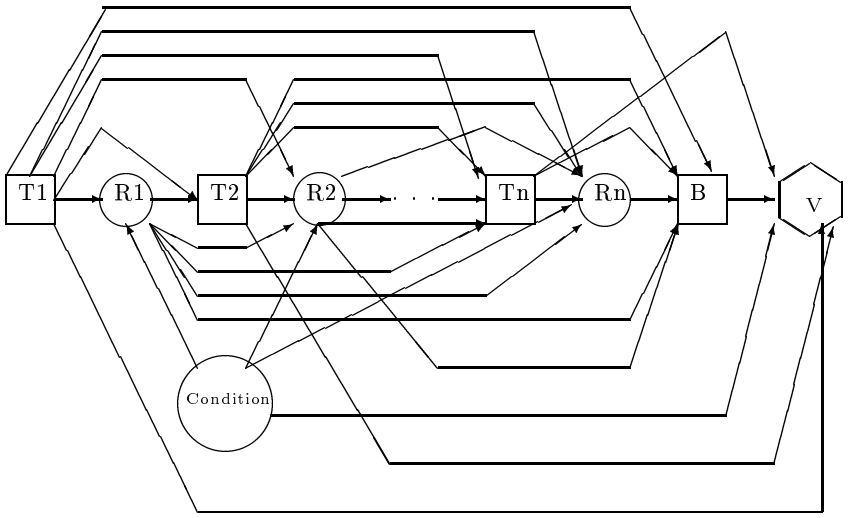

FIGURE 5.5: Influence diagram for the Generalized Buying Problem.

by using our heuristic against the same cost if the symmetric version of the problem is solved by OFP Heuristic (Li & D'Ambrosio, 1994).

It is assumed that final decision node B has a dimension of size 3 and the Condition node has a binary domain. All test decision nodes T_is and result nodes R_is are assumed to have dimensions of size 4 in our problem instances. In the first part of our experiments we measured the savings obtained by our heuristic for the instances of the problem where number of asymmetric decisions T_is varies from 1 to 4. The numbers are shown in Table 5.1 and their corresponding graph is displayed in Fig. 5.6. The results show that our heuristic gains exponential savings in the number of asymmetric decisions when it is applied to this class of decision problems.

In the second part of experiments we measured the savings gained by our heuristic when the degree of asymmetry in decisions change. The degree of asymmetry in a relationship is called *savings factor* (Qi & Poole, 1995). For

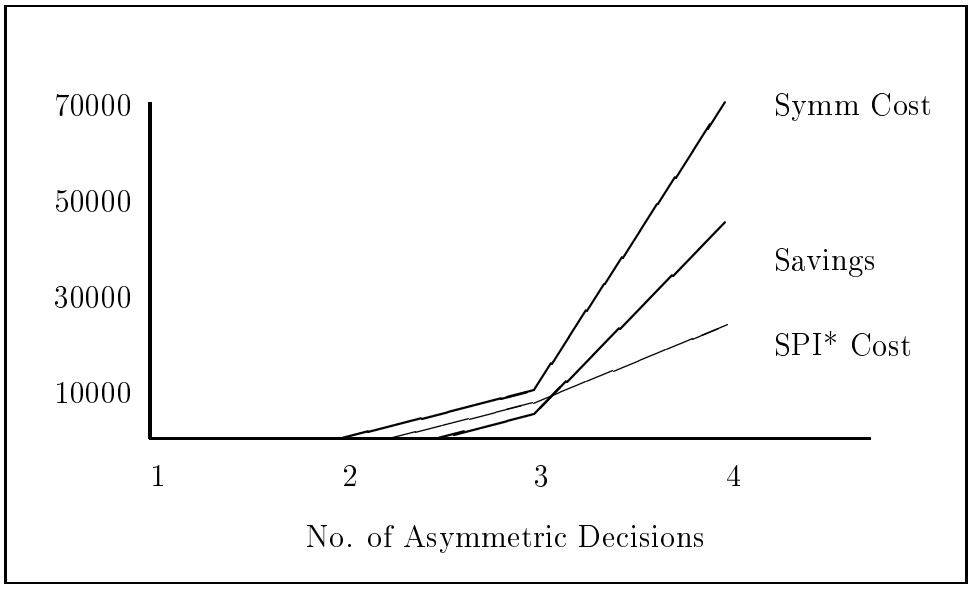

FIGURE 5.6: Savings and Number of decisions graph for Generalized Buying Problem.

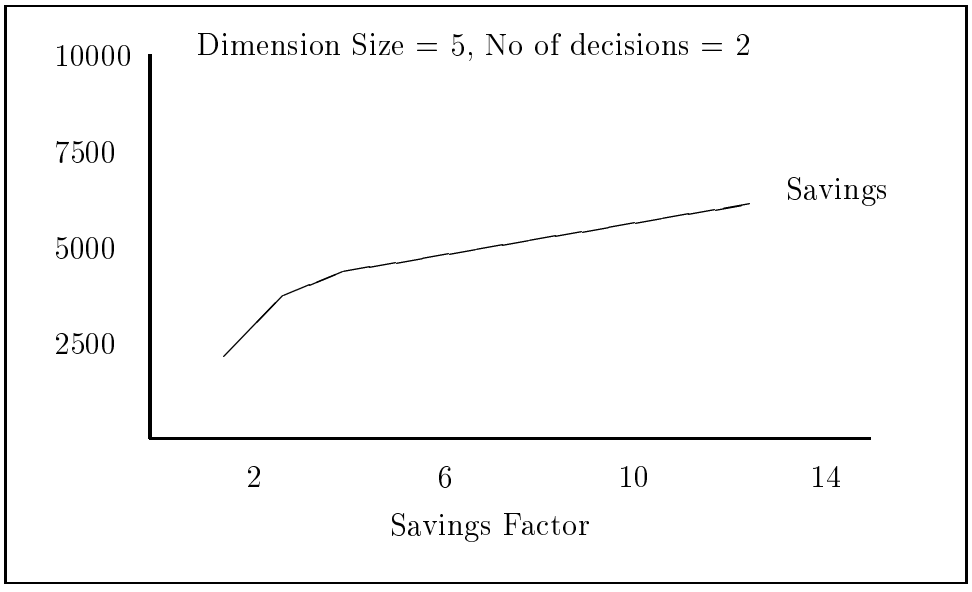

FIGURE 5.7: Savings and savings factor graph for Generalized Buying Problem.

No. of asym decisions	OFP-cost	SPI*-cost	Savings
1	108	103	5
2	900	874	26
3	8028	5841	2187
4	65619	27047	38572

TABLE 5.1: Savings vs the number of asymmetric decisions.

Decisions Savings Factor	OFP-cost	SPI*-cost	Savings
1.47	7225	5519	1706
2.5	7225	3617	3608
5	7225	2882	4343
12.5	7225	1165	6060

TABLE 5.2: Savings vs the savings factor.

decision node T_i with k_i alternatives and j_i possible outcom the savings factor is defined as $\frac{k_i * j_i}{h_i}$, where h_i is the number of points in the state-space of the relationship with non-zero probabilities. The number of asymmetric decisions and the number of their alternatives were 2 and 5, respectively, in all the problem instances. The savings are shown in Table 5.2. A corresponding graph is displayed in Fig. 5.7. The results show that our heuristic gains more savings if the asymmetric degree of relationships increases but such gains do not seem to be exponential.

5.6.2 Intercausal Independencies

A CPCS-type Example The following example in Fig. 5.8 is from (Zhang & Poole, 1996) and shows a small network where all children have ICI relationships with their parents. For this experiment we used the following local expressions with SPI* algorithm.

$$
\begin{aligned}
e(A) &= A_{t,f} \\
e(B) &= B_{t,f} \\
e(C) &= C_{t,f} \\
e(e1) &= 1_{e1_t|A_{t,f}} * 1_{e1_t|B_{t,f}} * 1_{e1_t|C_{t,f}} - e1_{e1_t|A_{t,f}} * e1_{e1_t|B_{t,f}} * e1_{e1_t|C_{t,f}} \\
&\quad + e1_{e1_f|A_{t,f}} * e1_{e1_f|B_{t,f}} * e1_{e1_f|C_{t,f}} \\
e(e2) &= 1_{e2_t|A_{t,f}} * 1_{e2_t|B_{t,f}} * 1_{e2_t|C_{t,f}} - e2_{e2_t|A_{t,f}} * e2_{e2_t|B_{t,f}} * e2_{e2_t|C_{t,f}} \\
&\quad + e2_{e2_f|A_{t,f}} * e2_{e2_f|B_{t,f}} * e2_{e2_f|C_{t,f}}
\end{aligned}
$$

$$
e(e3) = 1_{e3_t|e1_{t,f}} * 1_{e3_t|e2_{t,f}} - e3_{e3_t|e1_{t,f}} * e3_{e3_t|e2_{t,f}} + e3_{e3_f|e1_{t,f}} * e3_{e3_f|e2_{t,f}}
$$

A typical query for such a network is P(A/e3=t)=?, which is the probability of A given that e3 is observed to be true. SPI* found an evaluation for that query with 128 multiplications. A simulation of Heterogeneous Factorization (Zhang & Poole, 1996) shows that HF evaluation with the best variable elimination ordering takes 148 multiplications for the same query. If e3 is not observed then calculating P(A)=? takes 149 multiplications in SPI* while it takes 194 multiplications in HF.

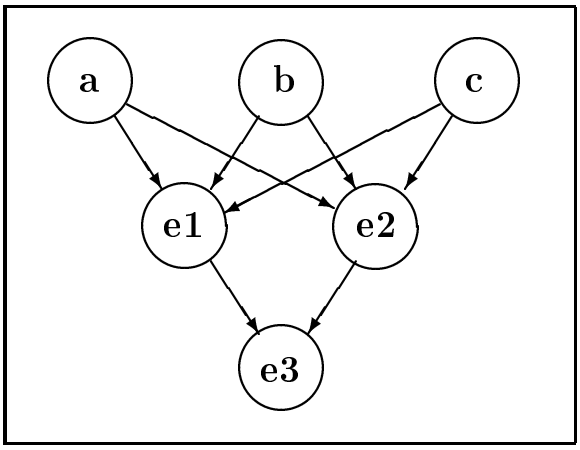

FIGURE 5.8: An Example CPCS-type Bayesian network.

5.6.3 Complexity

The time complexity of the algorithm is measured in the number of expressions relevant to the current query. If there are n expressions in the problem the complexity of candidate generation step is $O(4n^2)$. The cost evaluations for candidates are done in $O(K^2n)$ where K is a factor representing the number of terms in expressions. Therefore the time complexity of the algorithm is $O(4K^{n-1}n^3)$. However, it must be noted that such exponentiality does not automatically follow from having more than one term in expressions. For example, there are many situations where there n terms in expressions, but with disjoint state-spaces such that the term-by-term product of those expressions will have a number of terms, such as kn which is proportional to the number of terms in operand expressions, rather than an n^2. Also note that in practice we can see that the exponential part of our complexity is compensated by the huge savings obtained in the query evaluation costs in the number of real multiplications. (Qi & Poole, 1995) shows the exponential complexity of such

savings in the context of decision problems with asymmetries. On the other hand, it has been shown that all known special inference algorithms for ICI situations to date have worst-case exponential complexity (Dechter & Rish, 1998). The efficiency issues in relation with ICI and noisy-or situations are extensively discussed in Chapter Six.

Chapter 6

INTERCAUSAL INDEPENDENCIES IN BAYESIAN NETWORKS

Local parent-child or cause-effect relationships in Bayesian networks constitute an important part of both knowledge engineering and inference. In many situations individual causes influence an effect independent of one another. Such "Intercausal Independencies" (ICI) provide an opportunity to represent the local structures more efficiently and ease both the knowledge acquisition and the inference. Noisy-Or interaction (Pearl, 1988) (Srinivas, 1993) is one of the best studied and most widely used models of ICI. Different approaches have been proposed to represent ICI and to integrate the corresponding models into standard Bayesian network inference. Local Expression Languages (D'Ambrosio, 1995) provide a comprehensive approach for integration of many local structure models including ICI into standard Bayesian networks. Heterogeneous Factorization (Zhang & Poole, 1996) (Zhang & Poole, 1994) and temporal Belief Networks (Heckerman & Breese, 1994) are two other major approaches which are capable of representing many forms of causal independencies.

In this chapter these major approaches for representing independencies among causal effects in Bayesian networks will be compared. Some of the factors that influence the performance of the approaches in different situations will be identified. Finally, a new and better representation for intercausal independencies will be introduced.

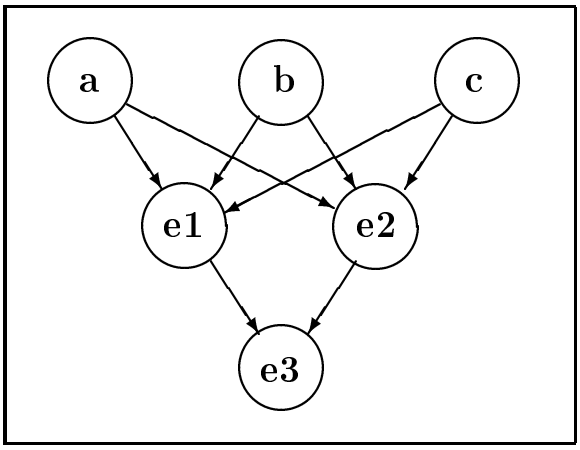

FIGURE 6.1: A Bayesian network for studying heterogeneous factorization.

6.1 Heterogeneous Factorization

Heterogeneous Factorization (HF) (Zhang & Poole, 1996) (Zhang & Poole, 1994) provides a way for Bayesian networks to incorporate intercausal independencies in the local structures of the network. Influences of the causes on an effect in ICI situations are represented in terms of individual factors in HF. Such effect variables are called "convergent" variables. For every convergent node a regular node is added to the network as its child, and this child is called the deputy of the corresponding convergent node. Fig. 6.1 shows an example Bayesian network where all the effect nodes are convergent. Fig. 6.2 shows the deputation network that is used in HF for inference.

As the name implies, the independent individual influences of the causes are to converge at a convergent effect node. Once such a convergence occurs the corresponding factors are combined. Then the effect node is converted to a regular node through combination with its deputy, since there is no more need to keep the convergence mechanism at the effect node.

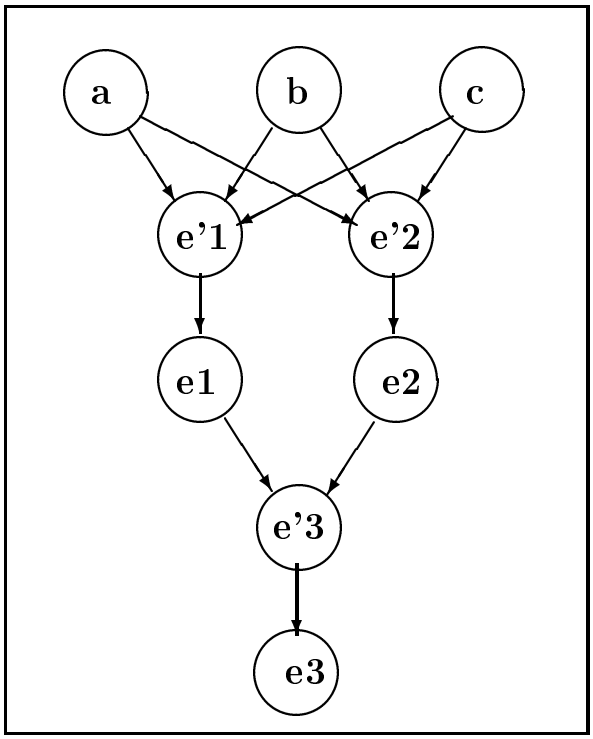

FIGURE 6.2: Deputation Bayesian network for Fig. 6.1 network.

The inference computation for standard Bayesian networks usually combines homogeneous probabilistic distributions uniformly through multiplication and addition operations. HF generalizes the combination operation so that the new heterogeneous individual factors can be combined with standard distributions during the inference. HF inference starts by finding an elimination ordering for those variables that are not included in the query. To eliminate a variable, all distributions and factors involving the variable are combined, and the resultant distribution is marginalized with regard to that variable.

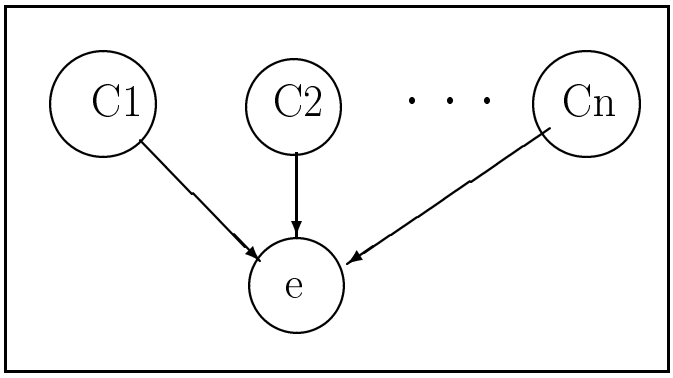

FIGURE 6.3: A Bayesian network for a multiple cause situation.

6.2 Temporal Transformation

Heckerman provides a new temporal definition of ICI (Heckerman, 1993). In the relationship between a cause and an effect, this model states the following. If the cause makes a transition from one of its states to another between times t and $t+1$ then the status of the effect at time $t+1$ depends only on the status of the effect at time t and the transition of the cause.

This model results in a transformation of a subgraph of a set of causes and an effect into a new subgraph. In the new subgraph the old effect is replaced by as many nodes as the number of the causes for the effect. Each of these new effect nodes represents the effect when only some of the causes have made their transitions. Fig. 6.3 shows a Bayesian network consisting of an effect that has a set of causes. Fig. 6.4 shows the network in Figure 4 after temporal transformation is performed on it. TT results in a standard Bayesian network where any standard algorithm may be used for inference. Note that transformation does not determine an ordering among parents when their corresponding effect-node instances are being added to the network.

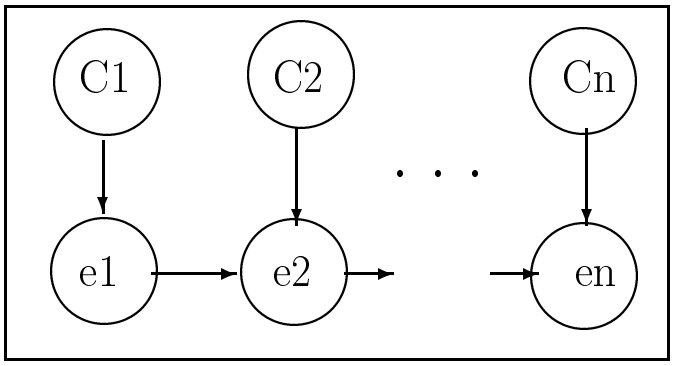

FIGURE 6.4: A temporal transformation of the network in Fig. 6.3.

6.3 A Look at the Performance of the Approaches

Local Expressions

The ESPI algorithm was introduced in Chapter Two. It tries to find the best order of operations on local expressions by successively selecting and combining candidate expressions whose combination will have the smallest set of remaining variables. Combination is followed by immediate reduction of the expression to a distribution. For example we mentioned earlier that combining expressions for nodes A and D in Fig. 2.1 gives,

$$\begin{aligned} exp(AD) \;=\; & 1_{D_t} * A_t - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} * A_t \\ & + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}} * A_t \\ & 1_{D_t} * A_f - c'_{D_t|A_{t,f}} * c'_{D_t|B_{t,f}} * A_f \\ & + c'_{D_f|A_{t,f}} * c'_{D_f|B_{t,f}} * A_f \end{aligned}$$

The algorithm at this point performs all the relevant products inside the expression and essentially reduces the expression to a distribution. For example, we have the $1_{D_t} * A_t$ product in the first term. Now one could ask the following

question. Is it not better to wait until we combine this expression with the expression for B and then decide about the order of multiplications among 1_{D_t}, A_t and B_t in the first term? In general, if we unnecessarily wait until more expressions are combined the number of terms will rapidly grow, rendering the cost high. On the other hand, if the products are performed too early, as it is currently done in ESPI, we may have to carry an unnecessarily large intermediate result inside the terms and thus rendering the cost even higher.

Heterogeneous Factorization

In HF the cost of combination operation in terms of real multiplications is exponential in the number of convergent variables that are shared among the factors to be combined. HF tries to avoid this by: (1) adding a deputy variable for every convergent variable in the network, and (2) constraining the elimination ordering so that heterogeneous factors for any convergent variable are combined into a standard distribution before being combined with other factors. The overhead cost of including the deputy variables in the inference, as well as the fact that the elimination ordering can not be always constrained to avoid situations mentioned in (2) can render high computation costs for some queries.

An Elimination ordering is usually found using Minimum Deficiency Ordering (MDO). MDO is well known as a heuristic triangulation method (Kjaerulff, 1990). So it is not unreasonable if we say that computation for the HF is concentrated in cliques of the triangulated version of the network. But note that HF algorithm does not require that the graph be moralized by connecting the unconnected parents of nodes prior to triangulation, and this is contrary to the requirements of most clique-tree oriented Bayesian network algorithms. This may create problems for regular parents-child relationships that may exist in

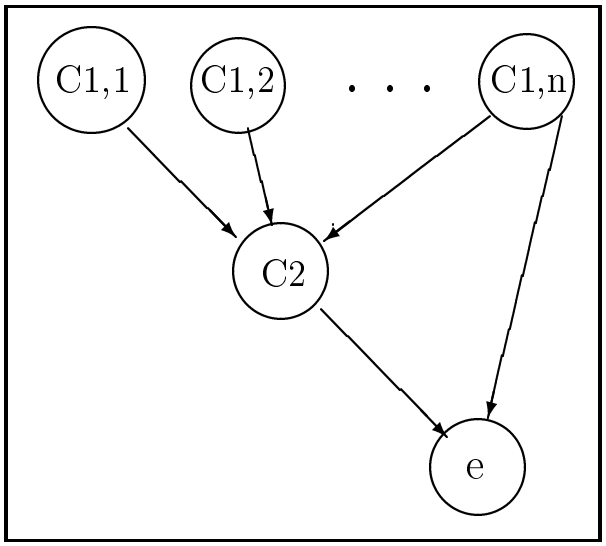

FIGURE 6.5: An example network for temporal transformation.

a network. See (Zhang, N. L. & Yan L., 1997) for related problems when it is attempted to use HF modeling with clique-tree propagation algorithm.

In practice, triangulation techniques can take into account the different state space sizes of the nodes and try to minimize the heaviest clique in the junction tree of the network (Becker, A. & Geiger, D., 1996). This is based on the following fact. The product of all state space sizes in a clique is proportional to the number of multiplications required to combine the distributions in the clique. This is true when the distributions are homogeneous and combinable through multiplication. However, with HF modeling it is not obvious at all what the effective state space sizes of convergent nodes are. Unless this deficiency is addressed, the elimination ordering algorithms such as MDO can not be directed toward finding an optimal ordering for the maximum efficiency in inference with HF in more general settings.

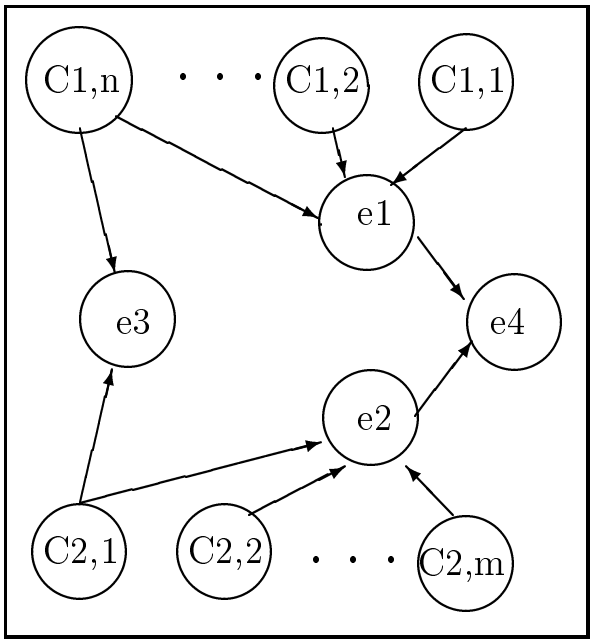

FIGURE 6.6: Another example network for temporal transformation.

Temporal Transformation

TT modeling in general produces a more factorized representation of local parents-child distributions. However, in doing so it adds many nodes to the network, which may make the network maintenance a difficult job in terms of memory requirements.

Depending on the structure of the original Bayesian network, different ordering among parents during TT may produce different networks with different structural properties. This in turn may render inference computations with different costs for the same original network.

One of the structural properties in Bayesian networks is the existence of loops in the undirected graph of the network. Such loops have some adverse effects on inference efficiency, because they prevent reduction of the size of

intermediate results during the stages of computations. Roughly speaking, the added cost of the inference is proportional to the diameters (or lengths) of such loops. If such a loop in the original network includes parent and child nodes of an ICI situation, TT modeling will result in a network that may have a corresponding loop with a higher diameter. For example, in the network in Fig. 6.5, suppose that ICI relationships exist between $C2$ and its parents, while e and its parents have a standard relationship. Then depending on the ordering among parents of $C2$ possible temporal transformations of the network can have loops whose sizes range from 3 to $n + 2$. Fig. 6.6 shows another example network for which TT may produce loops whose sizes range from 7 to $n + m + 5$ depending on the ordering among the parents of $e1$ and parents of $e2$. In this network $e4$ is assumed to have a standard relationship with its parents.

6.4 A Multiplicatively Factored Representation

We have seen that each of the above representations has certain disadvantages: the additive local expression language decomposition suffers from necessity to distribute expressions over the ICI child expressions. Heterogeneous factorization has costly overhead and constraints. The temporal transformation imposes an arbitrary ordering among ICI parents.

In this section we present a new, multiplicatively factored local expression language representation. Consider a noisy-or node with a set of parents and one child. We introduce an auxiliary variable for every parent so that we can separate the influences of individual parents on the child. These auxiliary variables play the same role the child instances variables play in TT, but they are not represented at the graphical level as individual nodes. We can accomplish the same functionality by including a local expression for every such variable in the

original child node expression. To provide interaction among such variables, TT imposes an order on parents so that auxiliary variables form a fixed chain. We, on the other hand, provide a flexible coupling among the expressions for auxiliary variables so that they can be combined in any order.

Consider $V = \{v_0, v_1, \ldots, v_n\}$ as a reference set and let $V^{*V'}$ denote any v_i such that $v_i \notin V'$ and $V' \subset V$. When sets such as V' are small, we abbreviate the notation by mentioning the individual elements of the sets. For example if $V' = \{v_j\}$ then we use V^{*v_j} for $V^{*V'}$. The following equalities are useful.

$$\begin{aligned} V^{*v_j} \cap v_i &= v_i, \quad i \neq j \\ &= \emptyset, \quad i = j \end{aligned}$$

$$\begin{aligned} V^{*v_j} \cap V^{*v_i} &= V^{*v_j v_i}, \quad i \neq j \\ &= V^{*v_j}, \quad i = j \end{aligned}$$

$$\begin{aligned} V^{*v_j} \cup V^{*v_i} &= V, \quad i \neq j \\ &= V^{*v_j}, \quad i = j \end{aligned}$$

Every variable v_i has a corresponding state-space that is denoted by v_{i_d}. For the time being, we further assume that $\forall i, j \;\; v_{i_d} = v_{j_d} = \{d0, d1, \ldots, dm\}$. $v_{i_{dk}}$ denotes an element in the state-space of v_i where v_i assumes the state dk. Now we let $V_{dk}^{*V'}$ denote any $v_{i_{dk}}$ such that $v_i \notin V'$ and $V' \subset V$. The following equality is very useful.

$$\begin{aligned} V_{dk}^{*v_j} \cap V_{dl}^{*v_i} &= V_{dk}^{*v_j, v_i}, i \neq j, dk = dl \\ &= \emptyset, \quad i \neq j, dk \neq dl \end{aligned}$$

Now, for our Fig. 2.1 example where A and B are ICI parents of D, we use auxiliary variables DA and DB and write the following local expressions.

$V = \{DA, DB\}$ is the only reference set in this situation. The joint probability table P(ABD) can be obtained by finding the conformal product of the above expressions, and marginalizing with regard to the auxiliary variables.

$$\begin{aligned} e(DA) &= 1_{DA_t|A_{t,f}V_t^{*DA}} + (1-c')_{DA_t|A_{t,f}V_f^{*DA}} + c'_{DA_f|A_{t,f}V_f^{*DA}} \\ e(DB) &= 1_{DB_t|B_{t,f}V_t^{*DB}} + (1-c')_{DB_t|B_{t,f}V_f^{*DB}} + c'_{DB_f|B_{t,f}V_f^{*DB}} \\ e(A) &= A_{t,f} \\ e(B) &= B_{t,f} \end{aligned}$$

In the ordering for the combination of expressions we assume that the corresponding algorithm treats the first expression of a set of auxiliary variables as an entry point and remove its coupling terms and dimension. Furthermore, the last of such expressions will be assumed to be treated as an exit point, where instances of corresponding auxiliary variable will be replaced by the corresponding instances of the original child variable. For example, if $e(DA)$ was an entry point it would be modified to,

$$e(DA) = (1-c')_{DA_t|A_{t,f}} + c'_{DA_f|A_{t,f}}$$

In a sense the above representation provides all potential advantages of the TT linear transformation approach, but it leaves the ordering aspects to be decided by the inference algorithm in an optimizing fashion.

6.5 New Representation and Inference Algorithms

V^* is a new construct for the local expression languages, and hence existing algorithms for inference with local expressions need some enhancements to handle the new construct. This section discusses the issues related to implementation of the new construct and shows that existing local expression algorithms need only little modifications to adapt the new construct. Our discussion will be focused on the changes to the factoring heuristic. Since the heuristic presented in Chapter Five is an extension of the factoring heuristic, it can adapt the new construct with similar modifications.

The new representation has an important property that facilitates its use in inference with factoring algorithms (Li & D'Ambrosio, 1994). The terms in an expression for noisy-or nodes in the new representation all have similar structures so that all of them can be abstractly represented with one characteristic term. Informally, we say an expression has a characteristic term if the following conditions are met: (1) All the terms have the same number of factors. (2) In any two terms, any factor in one term has exactly one and only one matching factor in the other term. The matching factors have identical state-spaces for the parent variables of the expression variable.

The following table shows some expressions and the state-space of their corresponding characteristic terms. The reader can easily verify that if V^*s are removed from such state-spaces then the problem of combining such state-spaces is a factoring problem.

Expression	Char term state-space
$A_{t,f}$	$A_{t,f}$
$1_{Ea_t/A_{t,f}V_t^{*Ea}} + (1-c')_{Ea_t/A_{t,f}V_f^{*Ea}} + c'_{Ea_f/A_{t,f}V_f^{*Ea}}$	$Ea_x/A_{t,f}V_x^{*Ea}$

To apply the optimal factoring heuristic there are two types of operational issues that have to be addressed. The first type is about how to perform set intersection and set size calculation with V^*s. The definitions provided for V^*s operations earlier completely specify the size calculation as well as intersection of the sets involving V^*s and hence they will not be discussed any further here.

The second type of operational issues is about how to marginalize over auxiliary variables specified with V^* notation. An auxiliary variable changes role to a regular variable in the following way. Any node expression can contain only one auxiliary variable from a reference set. When combining nodes for candidate generation If the nodes have both auxiliary variables then one of them must turn regular by stripping the term and dimension elements for V^* specifications. Then combination proceeds as usual. If such a candidate is selected as the next combination, the auxiliary variable turned regular must be removed from the corresponding reference set as well. If as a result of updating a reference set becomes empty, the last auxiliary variable will be marked as an exit auxiliary variable automatically.

An exit auxiliary variable turns regular if the corresponding reference set is empty. In general, and with few exceptions, either of the two auxiliary nodes can be turned regular. Therefore, two candidate combinations must be generated and their size evaluated. In some cases one of the two may be already designated as an exit, in which case only the other one will turn regular.

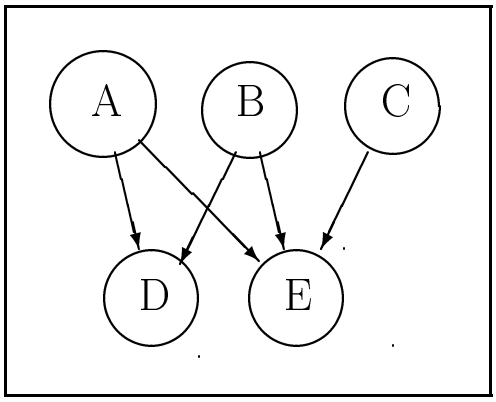

FIGURE 6.7: An example BN2O network.

When nodes contain auxiliary variables from different reference sets, we require that only exit nodes can be combined. If either of the two reference sets have no exit nodes up to this point, then the candidate nodes will be designated as exit nodes in case the candidate comb is selected.

We will demonstrate the above ideas for the example BN2O network shown in Fig. 6.7. Query is $P(B/D_tE_t)$.

The expressions and their corresponding characteristic term state-spaces are shown in Table 6.1. Note that in characteristic terms x can be either t or f but not both.

Cycle 1: (C, Ec) has a remaining size 2 and will be selected for combination. This is denoted as CEc, and after we sum C out we denote it with Ecc. Remaining size for the candidate (A, Da) is 3, which is for variables A, Da, and V^{*Ea}. For (Da, Db) the remaining size is 5. This is for variables Da, A, V^{*Ea} in Da, V^{*Eb} in Db, and B. However, note that combination will unite V^{*Ea} and V^{*Eb} into one, since they are from the same reference set.

Cycles 2 and 3: Assume that A and Da as well as B and Eb will be

Expression	Char term state-space
$A_{t,f}$	$A_{t,f}$
$B_{t,f}$	$B_{t,f}$
$C_{t,f}$	$C_{t,f}$
$1_{Da_t/A_{t,f}V_t^{*Ea}} + (1-c')_{Da_t/A_{t,f}V_f^{*Ea}} + c'_{Da_f/A_{t,f}V_f^{*Ea}}$	$Da_x/A_{t,f}V_x^{*Ea}$
$1_{Db_t/A_{t,f}V_t^{*Eb}} + (1-c')_{Db_t/A_{t,f}V_f^{*Eb}} + c'_{Db_f/A_{t,f}V_f^{*Eb}}$	$Db_x/A_{t,f}V_x^{*Eb}$
$1_{Ea_t/A_{t,f}V_t^{*Ea}} + (1-c')_{Ea_t/A_{t,f}V_f^{*Ea}} + c'_{Ea_f/A_{t,f}V_f^{*Ea}}$	$Ea_x/A_{t,f}V_x^{*Ea}$
$1_{Eb_t/A_{t,f}V_t^{*Eb}} + (1-c')_{Eb_t/A_{t,f}V_f^{*Eb}} + c'_{Eb_f/A_{t,f}V_f^{*Eb}}$	$Eb_x/A_{t,f}V_x^{*Eb}$
$1_{Ec_t/A_{t,f}V_t^{*Ec}} + (1-c')_{Ec_t/A_{t,f}V_f^{*Ec}} + c'_{Ec_f/A_{t,f}V_f^{*Ec}}$	$Ec_x/A_{t,f}V_x^{*Ec}$

TABLE 6.1: Characteristic terms for the expressions in example BN2O network

combined. This will reduce our factor set elements to, DaA, Db, Ea, EbB, and Ecc.

Cycle 4: (DaA, Db) has a remaining size 3. (Ea, Ecc) and (EbB, Ecc) both have the same remaining size if Ecc can be summed out (with regard to duplicity of candidate generation mentioned earlier). The tie breaker is the original aggregate size of factors which is 6 for (DaA, Db) and 4 for the other candidates. So (DaA, Db) will be the selected combination candidate, and it will be marginalized with regard to Da.

The result denoted by DbA is an exit as well as a regular node by definition. Due to being an exit node, Db will be instantiated to t in the expression at this point. Also note that when we calculate the remaining size for (DaA, Db) the fact that their reference set will be diminished is taken into account.

It is also noteworthy that (DaA, Ea) has the remaining size 4. If such a candidate is selected as the top candidate in a situation, then we see that DaA, which is an auxiliary node, will be combined with another auxiliary node from a different reference set. That combination takes place prior to DaA combination with other auxiliary nodes from its reference set. In that situation it is required that both nodes will be designated the exit nodes for their corresponding reference sets. This clearly shows the potentials that are provided by the dynamic ordering of auxiliary nodes versus the static and non-query oriented ordering among them.

Cycle 5: (Ea, Ecc) will be selected for combination. If Ecc is treated as a regular variable, Then Ecc is summed out and Ea remains.

Cycle 6: (Ea, EbB) will be selected for combination. The exit node designation and the corresponding instantiation take place here.

Cycle 7: $(EaEbB, DbA)$ will be our last combination that lets us marginalize with regard to A.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In conclusion this thesis has established the following points.

- Probabilistic inference with local expressions for non-standard Bayesian networks is a combinatorial optimization problem.

- Based on an analogy with optimal search algorithms such as A*, an effective heuristic solution called SPI* for that combinatorial optimization problem has been developed. SPI* is an inference algorithm which is capable of handling many non-standard local interactions more efficiently than previous algorithms for the same tasks.

- Local expression language has been in existence for about a decade. This thesis is the first to show a reasonable approach as well as an analysis on how to verify the coherency of such expressions.

- There have been many efforts trying to integrate graph-based Bayesian network inference algorithms with non-standard local interaction models. This thesis is the first to propose an effective measure of weight for nodes with non-standard distributions. The proposed measure can be applied in graph-based approaches such as variable elimination and clique-based algorithms.

- By developing new constructs for local expression language such as V^* we showed that value set algebra and its representation in local expression language are the best potential and least restrictive modeling mechanism for finer-grain representation of local structures in Bayesian networks.

7.2 Future Work

The following is a listing of interesting issues and questions that are still open for further investigation.

- The complexity of the value sets join chain optimization problem is still unknown. This problem has a subproblem called OFP, whose complexity is still unknown as well. We believe both problems are NP-Hard, but we have not proven that for either problem.

- There has been no comprehensive review and comparison of the inference approaches for Bayesian networks with asymmetric relationships. The partial studies that were done lack detailed analytical and experimental data and results.

- SPI* heuristic strategies for symbolic computations are rather simple. More studies are needed to refine such strategies and bring them closer to optimal.

- From what we described about the implementation of the V^* construct, it must be obvious that it will definitely reduce the complexity of the numerical computations for inference. But its impact on the complexity of the symbolic computations still need to be determined through experiments.

BIBLIOGRAPHY

Becker, A., & Geiger, D. (1996). "A Sufficiently Fast Algorithm for Finding Close to Optimal Junction Trees". In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 81–89.

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). "Context-Specific Independence in Bayesian Networks". In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 115–123.

Dagum, P., & Galper, A. (1993). "Additive Belief-Network Models". In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 91–98.

D'Ambrosio, B. (1990). Symbolic Probabilistic Inference in Belief Networks, Technical Report 90-30-1. , Dept. of Computer Science, Oregon State University.

D'Ambrosio, B. (1995). "Local Expression Languages for Probabilistic Dependence". *Intl Jrnl of Approximate Reasoning*, *Vol 13*(No 1), 61–81.

Dechter, R., & Rish, I. (1998). On the impact of causal independence, ICS Technical Report, October 1998. , Dept. of Information and Computer Science, University of California, Irvine.

Fung, R., & Shachter R. (1991). Contingent Influence Diagrams. Submitted for publication.

Geiger, D., & Heckerman D. (1991). "Advances in Probabilistic Reasoning". In *Proceedings of the Seventh Annual Conference on Uncertainty in AI*, pp. 118–126.

Geiger, D., Verma, T., & Pearl, J. (1989). "d-separation: From theorems to algorithms". In *Proceedings of the Fifth Workshop on Uncertainty in AI*, pp. 118–125.

Heckerman, D. (1989). "A Tractable Inference Algorithm for Diagnosing Multiple Diseases". In *Proceedings of the Fifth Workshop on Uncertainty in AI*, pp. 174–181.

Heckerman, D. (1991). *Probabilistic Similarity Networks.* MIT Press, Cambridge, MA.

Heckerman, D. (1993). "Causal Independence for Knowledge Acquisition and Inference". In *Proceedings of the Ninth Annual Conference on Uncertainty in AI*, pp. 122–127.

Heckerman, D., & Breese, B. (1994). "A New Look at Causal Independence". In *Proceedings of the Tenth Annual Conference on Uncertainty in AI*, pp. 286–292.

Henrion, M. (1988). *"Practical Issues in Constructing a Bayes Belief Network"*, pp. 132–139. North Holland, Amsterdam.

Howard, R. A. (1989). *"The used car buyer"*, Vol. II, pp. 689–718. Strategic Decision Group, Menlo Park, CA.

Kjaerulff, U. (1990). "Traingulation of Graphs-algorithms giving small total state space.". Tech. rep., R 90-09, Department of Mathematics and Computer Science, Aalborg University.

Li, Z., & D'Ambrosio, B. (1994). "Efficient Inference in Bayes Nets as a Combinatorial Optimization Problem". *Intl Jrnl of Approximate Reasoning, Vol 11* (No 1), 55–81.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, San Mateo, CA.

Pradhan, M., Provan, G., Middleton, B., & Henrion, M. (1994). "Knowledge Engineering for Large Belief Networks". In *Proceedings of the Tenth Annual Conference on Uncertainty in AI*, pp. 484–490.

Qi, R., & Poole, D. (1995). "A New Method for Influence Diagram Evaluation". *Computational Intelligence*, 498–528.

Raiffa, H. (1968). *Decision Analysis.* Addison Wesley.

Shachter, R. D., & Peot, M. A. (1992). "Decision Making Using Probabilistic Inference Methods". In *Proceedings of the Eighth Conference on Uncertainty in AI*, pp. 276–283.

Shenoy, P. P. (1996). *"Representing and Solving Asymmetric Decision Problems Using Valuation Networks"*, pp. 99–108. Lecture Notes in Statistics 112. Springer-Verlag.

Smith, J. E., Holtzman, S., & Matheson, J. E. (1993). "Structuring Conditional Relationships in Influence Diagrams". *Operations Research, Vol 41* (No 2), 280–297.

Srinivas, S. (1993). "A Generalization of the Noisy OR Model". In *Proceedings of the Ninth Annual Conference on Uncertainty in AI*, pp. 208–215.

Zhang, N. L., & Poole, D. (1994). "Intercausal Independence and Heterogeneous Factorization". In *Proceedings of the Tenth Annual Conference on Uncertainty in AI*, pp. 606–614.

Zhang, N. L., & Poole, D. (1996). "Exploiting Causal Independence in Bayesian Network Inference". *Journal Of Artificial Intelligence Research*, 301–328.

Zhang, N. L., & Yan L. (1997). "Independence of Causal Influence and Clique Tree Propagation". In *Proceedings of the Thirteenth Annual Conference on Uncertainty in AI*, pp. 481–488.