On the Role of Context-Specific Independence in Probabilistic Inference

Nevin L. Zhang
Department of Computer Science
Hong Kong University of Science & Technology
Clear Water Bay Road, Kowloon, Hong Kong
lzhangccs.ust.hk

David Poole
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada
pooleocs.ubc.ca

Abstract

Context-specific independence (CSI) refers to conditional independencies that are true only in specific contexts. It has been found useful in various inference algorithms for Bayesian networks. This paper studies the role of CSI in general. We provide a characterization of the computational leverages offered by CSI without referring to particular inference algorithms. We identify the issues that need to be addressed in order to exploit the leverages and show how those issues can be addressed. We also provide empirical evidence that demonstrates the usefulness of CSI.

1 Introduction

The theory of probabilistic inference begins with a joint probability over all variables of interest. The amount of numbers it takes to specify a joint probability is exponential in the number of variables. For this reason, probabilistic inference was thought to be infeasible until the introduction of Bayesian networks (BNs) [Pearl, 1988; Howard and Matheson, 1984]. Making use of conditional independence, a BN factorizes a joint a probability into a list of conditional probabilities. The factorization renders inference computationally feasible in many applications because each of the conditional probabilities involves only a fraction of the variables.

In practice, there are often conditional independence relationships that are true only in specific contexts. The concept of context-specific independence (CSI) was introduced specifically for such relationships. CSI has its roots in the influence diagram literature [Olmsted, 1983; Fung and Shachter, 1990; Smith *et* al., 1993) and was first formalized by [Boutilier *et* al., 1996]. Researchers have shown that CSI can be exploited to speed up various Bayesian network inference algorithms such as symbolic probabilistic inference [D'Ambrosio, 1994], search Santos Jr. and Shimony, 1996], cutset conditioning Boutilier *et* al., 1996; Geiger and Heckerman, 1996, clique tree propagation (CTP) [Boutilier *et al*., 1996], arc reversal [Cheuk and Boutilier , 1997], and variable elimination (VE) [Poole, 1997].

This paper results from efforts to identify the common principle underlying those works. We attempt to answer the following questions: Why in general CSI leads to faster inference? In other words, how do we characterize the computational leverages offered by CSI without referring to particular inference algorithms? What issues do we need to address in order to exploit the leverages? How do we address those issues? Finally, how much can we gain?

It is well known that the computational leverages afforded by conditional independence can be characterized in terms of factorization: conditional independence allows one to factorizes a joint probability into a list of conditional probabilities. As it turns out, the computational leverages offered by CSI can also be characterized in terms of factorization. More specifically, CSI allows one to further decompose some of the conditional probabilities, giving rise to a finer-grain factorization of the joint probability. This is precisely why CSI can speed up inference.

In order to take advantage of the finer-grain factorization, the main technical issue that one needs to address is that some of the factors in the factorization are partial functions. Fortunately, this issue can easily be addressed using an operation called union-product.

In addition to providing a clear picture about the role of CSI in probabilistic inference, this paper also gives a general method for exploiting CSI. The method can be easily grounded with popular inference algorithms such as CTP [Lauritzen and Spiegelhalter, 1988; Jensen *et* al., 1990; Shafer and Shenoy, 1990] and VE [Zhang and Poole, 1996; Dechter, 1996]. All one has to do is change one basic operation, namely replacing product of full functions with union-product of partial functions.

Experiments have been performed to empirically demonstrate the effectiveness of CSI. The results confirmed that CSI can significantly speed up inference.

2 Bayesian Networks and Probabilistic Inference

To start with, this section briefly reviews the concepts of Bayesian networks and factorization. We also explain

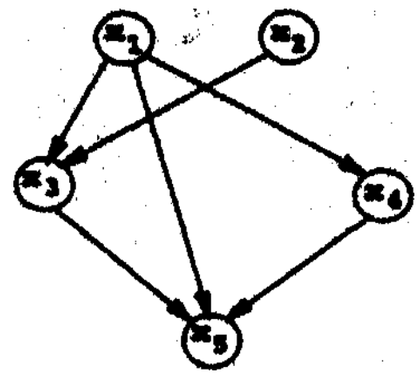

Figure 1: A Bayesian network.

why factorization is the key to efficient inference.

2.1 Bayesian Networks

A *Bayesian network* [1] (BN) is an annotated directed acyclic graph, where each node represents a random variable and is associated with the conditional probability of the node given its parents. In addition to the explicitly represented conditional probabilities, a BN also implicitly represents conditional independence assertions. Let $x_1, x_2, \ldots, x_n$ be an enumeration of all the nodes in a BN such that each node appears before its children and let π_{x_i} be the set of parents of a node xi. The following assertions are implicitly represented: Each variable xi is conditionally independent of variables in $\{x_1, x_2, \ldots, x_{i-1}\} \backslash \pi_{x_i}$ given variables in π_{x_i}.

The conditional independence assertions and the conditional probabilities $P(x_i|\pi_i)$ attached to the nodes together entail a joint probability over all variables. As a matter of fact, we have

$$\begin{aligned} & P(x_1, x_2, \ldots, x_n) \\ &= \prod_{i=1}^{n} P(x_i|x_1, x_2, \ldots, x_{i-1}) \\ &= \prod_{i=1}^{n} P(x_i|\pi_{x_i}), \end{aligned} \quad (1)$$

where the first equality follows from chain rule and the second follows from the conditional independence assertions.

2.2 Probabilistic Inference

Inference refers to the process of computing the posterior probability $P(X|Y=Y_0)$ of a list X of query variables after obtaining some observations $Y=Y_0$. Here Y is a list of observed variables and Y_0 is the corresponding list of observed values.

The posterior probability $P(X|Y=Y_0)$ can be obtained from the marginal probability $P(X,Y)$, which in turn can be computed from the joint probability $P(x_1, x_2, \ldots, x_n)$ by marginalizing out variables outside $X \cup Y$ one by one. Since a BN implicitly represents a joint probability, one can in theory perform arbitrary inference. In practice, this is not viable because marginalizing out a variable from a joint probability requires an exponential number of additions.

[1] Also known as probabilistic influence diagrams and belief networks.

The key to efficient inference lies in the concept of factorization. A *factorization* of a joint probability is a list of *factors* (functions) from which one can reconstruct the joint probability.

Because of (1), we say that a BN *factorizes* a joint probability $P(x_1, x_2, \ldots, x_n)$ into conditional probabilities $p(x_1|\pi_{x_1})$, $p(x_2|\pi_{x_2})$, ..., and $p(x_n|\pi_{x_n})$ and that the conditional probabilities constitute a *multiplicative factorization* of the joint probability. The BN in Figure 1, for instance, gives us the following multiplicative factorization of $P(x_1, x_2, \ldots, x_5)$:

$$P(x_1)P(x_2)P(x_3|x_1, x_2)P(x_4|x_1)P(x_5|x_1, x_3, x_4). \quad (2)$$

We will use this network as a running example through out the paper.

To see why factorization is of fundamental importance to inference, consider a joint probability $P(x_1, x_2, \ldots, x_n)$ over n binary variables. To marginalize out a variable x_i means to compute $\sum_{x_i} P(x_1, x_2, \ldots, x_n)$. This computation is *global* since all variables are involved. It takes 2^{n-1} numerical additions and hence is infeasible except when n is very small.

Now suppose we have a multiplicative factorization of the joint probability $\{f_1, f_2, \ldots, f_m\}$ and only the first k factors involve x_i. By distributivity, we have

$$\sum_{x_i} P(x_1, x_2, \ldots, x_n) = \prod_{j=k+1}^{m} f_j \sum_{x_i} \prod_{j=1}^{k} f_j.$$

Consequently, we can marginalize out x_i from the factorization as follows:

1. Remove from the factorization all functions that involve x_i;
2. Compute the product of the functions;
3. Marginalize out x_i from the product; and
4. Put the resulting function back to the factorization.

This is the principle underlying inference algorithms such as CTP and VE. Here one needs to compute $\sum_{x_i} \prod_{j=1}^{k} f_j$. This computation is *local* in the sense that it involves only some of the variables. It is usually much cheaper than the global computation mentioned above.

In our running example, marginalizing out x_2 means to compute $\sum_{x_2} P(x_1, x_2, x_3, x_4, x_5)$ without factorization. This takes 16 additions. With factorization, on the other hand, one needs to compute $\sum_{x_2} P(x_2)P(x_3|x_1, x_2)$, which takes only 4 additions.

3 CSI and Decomposition of Conditional Probabilities

We next review the concept of CSI and shows how it leads to decomposition of conditional probabilities.

3.1 Context-Specific Independence

Let C be a set of variables. A *context* on C is an assignment of one value to each variable in C. We denote a context by $C=\gamma$, where γ is a set of values of variables

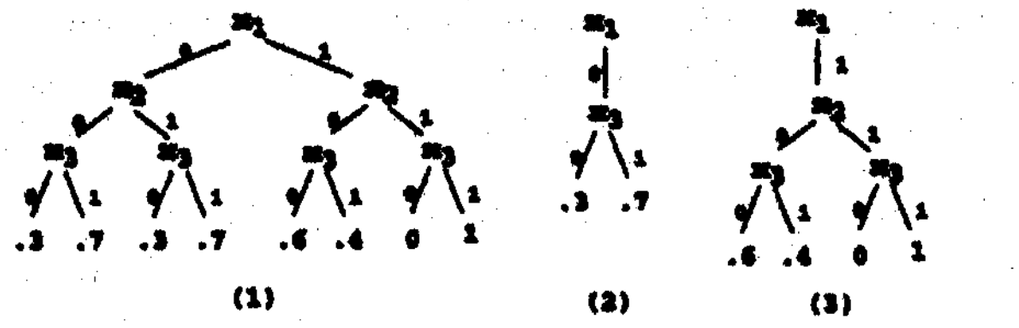

Figure 2: $P(x_3|x_1,x_2)$ and its decomposition.

in *C.* Two contexts are *incompatible* if there exists a variable that is assigned different values in the contexts. They are *compatible* otherwise.

This following definition of CSI is due to [Boutilier *et* al., 1996]. Let X, Y, Z, and *C* be four disjoint sets of variables. *X* and *Y* are *independent given Z in context* $C=\gamma$ if

$$P(X|Z,Y,C=\gamma) = P(X|Z,C=\gamma)$$

whenever $P(Y,Z,C=\gamma)>0$. When *Z* is empty, one simply says that *X* and *Y* are *independent in context* $C=\gamma$.

As an example, consider four variables: income, profession, weather, and qualification. A farmer's income depends on weather and typically does not depend on his qualification. On the other hand, a office clerk's income depends on his qualification and typically does not depend on weather. In other words, income is independent of qualification in the context "professionsfarmer" and it is independent of weather in the context "profession=office-clerk".

3.2 Decomposition of Conditional Probabilities

To illustrate how CSI leads to decomposition of conditional probabilities, we use $P(x_3|x_1,x_2)$ as an example. Consider $P(x_3|x_1=0,x_2)$ and $P(x_3|x_1=1,x_2)$ separately. Assume x_3 is independent of x_2 in context $x_1=0$. Then $P(x_3|x_1=0,x_2)=P(x_3|x_1=0)$. Consequently, we can decompose $P(x_3|x_1,x_2)$, which requires 8 numbers to specify, into two smaller components $P(x_3|x_1=0)$ and $P(x_3|x_1=1,x_2)$, which require only 6 numbers to specify.

To make the example more concrete, suppose $P(x_3|x_1,x_2)$ is given by the tree shown in Figure 2 (1). The tree states that $P(x_3=0|x_1=0,x_2=0)$, for instance, is 0.3. Because x_3 is independent of x_2 in context $x_1=0$, the tree can be decomposed into the two smaller trees shown in Figure 2 (2) and (3), which represent $P(x_3|x_1=0)$ and $P(x_3|x_1=1,x_2)$ respectively.

Next assume x_5 is independent of x_3 given x_4 in context $x_1=0$ and x_5 is independent of x_4 given x_3 in context $x_1=1$. Then $P(x_5|x_1=0,x_3,x_4)=P(x_5|x_1=0,x_4)$ and $P(x_5|x_1=1,x_3,x_4)=P(x_5|x_1=1,x_3)$. Consequently, we can decompose $P(x_5|x_1,x_3,x_4)$, which requires 16 numbers to specify, into two smaller components $P(x_5|x_1=0,x_4)$ and $P(x_5|x_1=1,x_3)$, which take only 8 numbers to specify. For concreteness, assume the two smaller components are given by the trees in Figure 3.

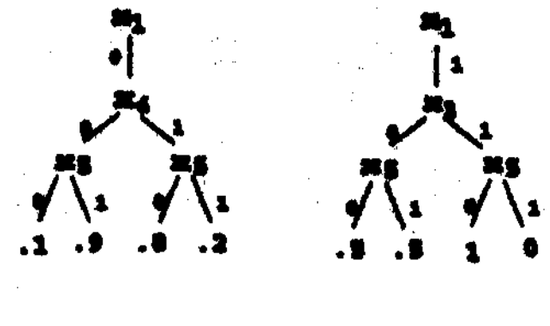

Figure 3: Decomposition of $P(x_5|x_1,x_3,x_4)$.

After the decompositions, the decomposition given in (2) becomes

$$P(x_1), P(x_2), P(x_3|x_1=0), P(x_3|x_1=1,x_2),$$
$$P(x_4|x_1), P(x_5|x_1=0,x_4), P(x_5|x_1=1,x_3) \quad (3)$$

This decomposition is of finer-grain because the conditional probabilities of x_3 and x_5 have be broken up into smaller pieces.

4 Making Inference with Refined Factorizations

This section shows how to make inference with factorizations such as the one given by (3). A technical issue that we need to address is that some of the factors are partial functions. For example, $P(x_3|x_1=0)$ is a partial function of x_1 and x_3 in the sense that it is not defined for the case when $x_1=1$.

In general, a *partial function* of a set *X* of variables is a mapping from a proper subset of possible values of *X* to the real line. In other words, it is defined only for some but not all possible values of *X.* The set of possible values of *X* for which a partial function is defined is called the *domain* of the partial function. A *full function* of *X* is a mapping from the set of all possible values of *X* to the real line. In other words, it is defined for all possible values of *X.* In the rest of a paper, we will use the term "function" when we are not sure whether a function is a partial function or a full function.

4.1 Union-Product

To manipulate partial functions, we need the operation of union-product. Suppose *X, Y,* and *Z* are three disjoint sets of variables and suppose $g(X,Y)$ and $h(Y,Z)$ are two functions. The *union-product* $g \cup h$ *of* g *and* h [2] is the function of variables in $X \cup Y \cup Z$ given by

$$(g \cup h)(X,Y,Z) = \begin{cases} \text{undefined} & \text{if } g(X,Y) \text{ \& } h(Y,Z) \text{ undefined} \\ g(X,Y) & \text{if } g(X,Y) \text{ defined, } h(Y,Z) \text{ undefined} \\ h(Y,Z) & \text{if } g(Y,Z) \text{ undefined, } h(X,Y) \text{ defined} \\ g(X,Y)h(Y,Z) & \text{if both } g(X,Y) \text{ \& } h(Y,Z) \text{ defined} \end{cases}$$

The operation is illustrated in Figure 4. We sometimes write $g \cup h$ as $g(X,Y) \cup h(Y,Z)$ to make explicit the arguments of / and *g.* When the domains of g and h are

[2] The notation $\cup$ is produced in LaTeX using macro {\cup}\hspace{-0.6em}{*}.

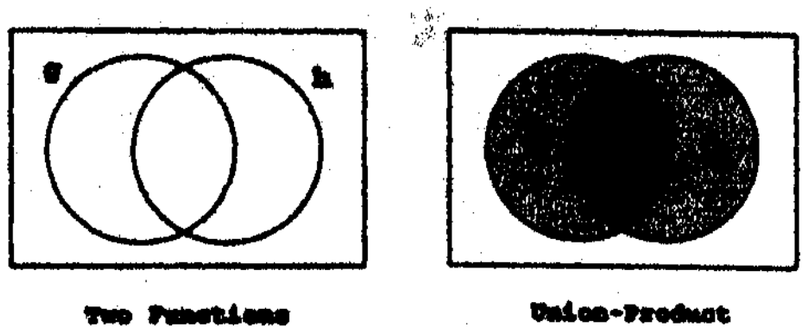

Figure 4: Union-product: The two circles in the left figure represent the domains of two functions *g* and *h*. The domain of the union-product $g \cup h$ is the union of those of *g* and *h*. The union-product equals the product of *g* and *h* in the area where both *g* and *h* are defined; it equals *g* in the area where only *g* is defined; and it equals *h* in the area where only *h* is defined.

disjoint, we call $g \cup h$ the union of *g* and *h* and write it as $g \cup h$.

Here are some of the properties of the union-product operation. First, the union-product of two full functions is simply their product. Together with the concept of union, this explains the term "union-product". Second, the union-product of a full function with another function, full or partial, is a full function. Third, the union-product operation is associative and commutative. We can hence talk about the union-product of a list of functions. The union-production of a list $\mathcal{F}$ of functions will be denoted as $\cup\mathcal{F}$.

4.2 Decompositions

The concept of union allows us to rigorously defined decomposition. A list $\mathcal{F}$ of functions with disjoint domains is *decomposition* of a function f if $f = \cup\mathcal{F}$. A decomposition is *proper* if no two functions in the decomposition share the same set of arguments. A decomposition of a function f is *nontrivial* if at least one function in the decomposition has fewer arguments than f itself. A function is *decomposable* if it has a nontrivial decomposition.

The function shown in Figure 2 (1) is decomposable. It can be nontrivially decomposed into the the two partial functions shown in Figure 2 (2) and (3).

4.3 Union-Product Factorizations

With union-product, we can now make explicit the sense in which the list of functions given in (3) is a factorization of the joint probability $P(x_1, x_2, x_3, x_4, x_5)$.

A list $\mathcal{F}$ of functions is a *union-product factorization,* or simply a *UP-factorization,* of a function f if $f = \cup\mathcal{F}$. Note that functions in a decomposition must have disjoint domains whereas domains of functions in a UP-factorization might intersect. A decomposition is a UP-factorization but not vice versa.

For any variable x, let $\mathcal{F}_x$ be the set of functions in $\mathcal{F}$ that contain x as an argument. A UP-factorization $\mathcal{F}$ is *normal* if $\cup\mathcal{F}_x$ is a full function whenever $\mathcal{F}_x \neq \emptyset$.

The list of function given in (3) is a factorization of the joint probability $P(x_1, x_2, x_3, x_4, x_5)$ because

$$
\begin{aligned}
& P(x_1) \cup P(x_2) \cup P(x_3|x_1=0) \cup P(x_3|x_1=1, x_2) \\
& \cup P(x_4|x_1, x_2) \cup P(x_5|x_1=0, x_4) \cup P(x_5|x_1=1, x_3) \\
= \; & P(x_1) \cup P(x_2) \cup [P(x_3|x_1=0) \cup P(x_3|x_1=1, x_2)] \\
& \cup P(x_4|x_1, x_2) \cup [P(x_5|x_1=0, x_4) \cup P(x_5|x_1=1, x_3)] \\
= \; & P(x_1) \cup P(x_2) \cup P(x_3|x_1, x_2) \cup P(x_4|x_1, x_2) \\
& \cup P(x_5|x_1, x_3, x_4) \\
= \; & P(x_1, x_2, x_3, x_4, x_5),
\end{aligned}
$$

where the first equality is true because the union-product operation is associative, the second equality follows from the assumptions made in Section 3, and the third equality follows from the first property of union-product. The factorization is also normal. For example, $\mathcal{F}_{x_3} = \{P(x_3|x_1=0), P(x_3|x_1=1, x_2), P(x_5|x_1=1, x_3)\}$. Since $P(x_3|x_1=0) \cup P(x_3|x_1=1, x_2) = P(x_3|x_1, x_2)$ is a full function, so must be $\cup\mathcal{F}_{x_3}$ by the second property of union-product.

In general, let $\mathcal{F}$ be the set that consists of, for each variable in a BN, the conditional probability of the variable or, when the conditional probability is decomposed, its components. Then $\mathcal{F}$ is a normal UP-factorization of the joint probability of all variables. It is of finer-grain than the multiplicative factorization given by the BN if at least one conditional probability is decomposed.

4.4 Inference with UP-Factorizations

The following theorem, which we state without proof, lays the foundation for making inference with normal UP-factorizations.

Theorem 1 *Suppose T is a normal UP-factorization of a full function f and x is an argument of f. Then $\mathcal{G} = (\mathcal{F} \setminus \mathcal{F}_x) \cup \{\sum_x \cup\mathcal{F}_x\}$ is a normal UP-factorization of $\sum_x f$.*

According to the theorem, a variable x can be marginalized out from a normal UP-factorization as follows:

1. Remove from the factorization all functions that involve x;
2. Compute the union-product of the functions;
3. Marginalize out x from the union-product; and
4. Put the resulting function back to the factorization.

Comparing this procedure with the one outlined in Section 2, we see that existing inference algorithms such as CTP and VE can be adapted to work with normal UP-factorizations by simply replacing product of full functions with union-product of partial or full functions.

The above descriptions are rather abstract. For example, implementations of union-product and marginalization are not given. Those and other details can be found in a longer version of the paper [Zhang, 1998]. In that paper, the method is also compared to previous methods.

5 CSI and Inference Efficiency

Using the running example, this section illustrates why inference with finer-grain UP-factorizations is more effi-

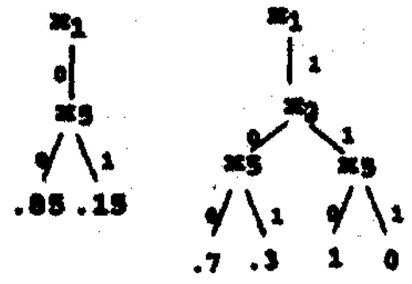

Figure 5: Elimination of variable x_3.

dent than with the corresponding multiplicative factorizations.

Consider marginalizing out x_3. Without CSI, we need to compute

$$\sum_{x_3} P(x_3|x_1,x_2)P(x_5|x_1,x_3,x_4). \quad (4)$$

With CSI, we need to compute

$$\sum_{x_3} P(x_3|x_1=0) \bowtie P(x_3|x_1=1,x_2) \bowtie P(x_5|x_1=1,x_3). \quad (5)$$

The second computation is cheaper than the first one for a number of reasons. The conditional probability $P(x_3|x_1,x_2)$ is decomposed into two components that takes fewer numbers to specify. The conditional probability $P(x_5|x_1,x_3,x_4)$ is also decomposed into two components that take fewer numbers to specify. Moreover, only one of those two components is involved in the second computation. As a consequence, the second computation has fewer variables (x_4 not involved) and fewer numbers to deal with.

In general, marginalizing out a variables from a finer-grain UP-factorization involves fewer variables and fewer numbers than with the corresponding multiplicative factorization. It is therefore cheaper.

There is another reason why finer-grain UP-factorizations leads to faster inference. The result of expression (5) is a full function of x_1 and x_5. The full function happens to be decomposable and can be decomposed into the two partial functions shown in Figure 5. In such a case, we can compute the decomposition directly. This is less expensive than computing the full function itself because the decomposition requires fewer numbers to specify.

When we compute decompositions of functions instead of functions themselves, we are *preserving structures.* Preserving structures not only benefits the current step of inference but also simplifies future steps. It is therefore an important issue. In [Zhang, 1998], this issue is addressed in detail.

6 Empirical Results

Experiments have been conducted to demonstrate the computational benefits of CSI. A BN named Water was used in the experiments [3]. Water is a model for the biological processes of a water purification plant. It consists

[3]Obtained from a Bayesian network repository at Berkeley.

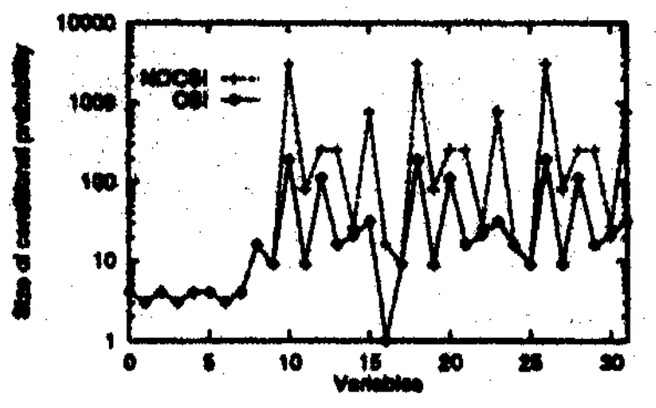

Figure 6: Representation complesdties of conditional probabilities with and without CSI.

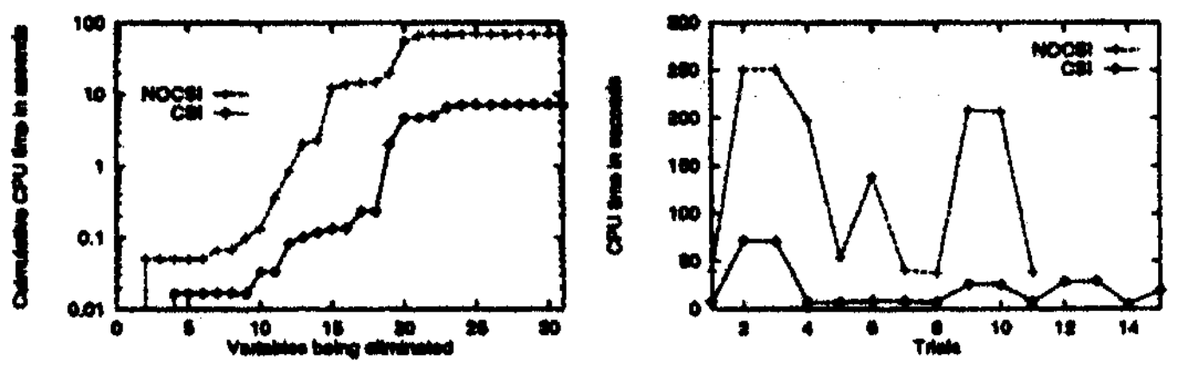

Figure 7: Performances of VE with and without CSI.

of 32 variables. Strictly speaking, conditional probabilities of the variables are not decomposable. To make them decomposable, some of the probability values were modified. The induced errors are upper bounded by 0.05. Using a decision-tree-like algorithm [Quintan, 1986], we were be able to decompose some of the modified conditional probabilities and thereby reduce their representation complexities drastically [4]. See Figure 6.

The experiments were based on the VE algorithm and were performed on a SUN ULTRA 1 machine. The task was to eliminate all variables according to a predetermined elimination ordering. In the first experiment, an ordering by Kjaerullf was used [5]. The the amounts of times in CPU seconds that VE took, with and without CSI, to eliminate the first n variables for n running from 0 to 31 are shown in left chart of Figure 7. We see that VE ran much faster with CSI. In particular, the entire elimination process took about 7 seconds with CSI. Without CSI, however, it took about 70 seconds.

In the second experiment, we generated 14 new elimination orderings by randomly permuting pairs of variables in Kjaerullf's ordering. A trial was conducted with each ordering. The performances of VE, with and without CSI, across all the trials are summarized in the right chart of Figure 7. We see that VE was significantly more efficient with CSI than without CSI. The speedup was more than one magnitude on average. Moreover, there are four trials where VE was not able to complete without CSI due to large memory requirements. With CSI, on

[4] Modifications of probability values actually take place during the decomposition process. We choose to describe them as two separate steps here for presentation clarity.

[5]The ordering was also obtained from the Berkeley repository.

the other hand, VE completed each of those four trials in less than 30 seconds.

7 Conclusions

This paper studies the role of CSI in Bayesian network inference. It differs from earlier work in that we do not attempt to demonstrate the usefulness of CSI in particular inference algorithms. Rather, we provide a general characterization of the computational leverages offered by CSI. This characterization fits well with the characterization of the computational leverages afforded by conditional independence. They both are in terms of factorization. While conditional independence provides one with a factorization of a joint probability, CSI allows one to refine the factorization.

We clearly identify the issues that one needs to address in order to takes advantage of the computational leverages offered by CSI. There are two issues: partial functions and preservation of structures. The first issue in addressed in detail and the second issue is addressed in [Zhang, 1998]. We also give a general method for exploiting CSI. The method can be easily grounded with popular inference algorithms such as CTP and VE. All one has to do is to replace product of full functions with union-product of partial and full functions. Finally, we provide empirical evidence that demonstrates the usefulness of CSI.

Acknowledgements

The authors thank the anonymous reviewers for useful comments and suggestions. Research is supported by Hong Kong Research Grants Council Grant HKUST6125/98E and Natural Sciences and Engineering Research Council of Canada Research Grant OGP0044121.

References

[Boutilier *et al,* 1996] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller, Context-specific independence in Bayesian networks, *Proc.of 12th Conference on Uncertainty in AI,* 115-123.

[Cheuk and Boutilier , 1997] A. Y. W. Cheuk and C. Boutilier, Structured arc reversal and simulation of dynamic probabilistic networks, *Proc. of 13th Conference on Uncertainty in AI,* 72-79.

[D'Ambrosio, 1994] B. D'Ambrosio, Local expression languages for probabilistic dependence, *Int. J. of Approximate Reasoning,* 11 (1), 1-16.

[Dechter, 1996] R. Dechter, Bucket elimination: A unifying framework for probabilistic inference, *Proc. of 12th Conference on Uncertainty in AI,* 211-219.

[Fung and Shachter, 1990] R. M. Fung and R. D. Shachter, Contingent Influence Diagrams, Advanced Decision Systems, 1500 Plymouth St., Mountain View, CA 94043, USA.

[Geiger and Heckerman, 1996] D. Geiger and D. Heckerman, Knowledge representation and inference in similarity networks and Bayesian multimets, *Artificial Intelligence, 92, 45~74.*

[Howard and Matheson, 1984] R. A. Howard, and J. E. Matheson, Influence Diagrams, *The principles and Applications of Decision Analysis,* Vol. II, R. A. Howard and J. E. Matheson (eds.). Strategic Decisions Group, Menlo Park, California, USA.

[Jensen *et al*., 1990] F. V. Jensen, K. G. Olesen, and K. Anderson, An algebra of Bayesian belief universes for knowledge-based systems, *Networks,* 20, 637 - 659.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter, Local computations with probabilities on graphical structures and their applications to expert systems, *Journal of Royal Statistical Society B,* 50: 2, 157 - 224.

[Olmsted, 1983] S. M. Olmsted, Representing and solving decision problems, Ph.D. Dissertation, Department of Engineering-Economic Systems, Stanford University.

[Pearl, 1988] J. Pearl, *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference,* Morgan Kaufmann Publishers, Los Altos, CA.

[Poole, 1997] D. Poole, Probabilistic partial evaluation: exploiting rule structure in probabilistic inference, *Proc. of 15th Int. Joint Conference on AI,* 1284-1291.

[Quinlan, 1986] J. R. Quinlan , Induction of decision trees, *Machine Learning,* 1, pp. 81-106.

[Santos Jr. and Shimony, 1996] E. Santos Jr. and S. E. Shimony, Exploiting case-based independence for approximating marginal probabilities, *Int. J. of Approximate Reasoning,* 14, 25-54.

[Shafer and Shenoy, 1990] G. Shafer and P. Shenoy, Probability propagation, *Annals of Mathematics and AI,* 2, 327-352.

[Smith *et al.,* 1993] J. E. Smith, S. Holtzman, and J. E. Matheson, Structuring conditional relationships in influence diagrams, *Operations Research,* 41, No. 2, 280-297.

[Zhang and Poole, 1996] N. L. Zhang and D. Poole, Exploiting causal independence in Bayesian network inference, *J. of AI Research,* 5, 301-328.

[Zhang, 1998] N. L. Zhang, Inference in Bayesian networks: The role of context-specific independence, Technical Report, HKUST-CS98-09, Department of Computer Science, University of Science and Technology, Hong Kong. Available at http://www.cs.ust.hk/lzhang/paper/csitr.html.