

Probabilistic Reasoning in Bayesian Networks: A Relational Database Approach

S.K.M. Wong, D. Wu and C.J. Butz

Department of Computer Science, University of Regina
Regina Saskatchewan, Canada S4S 0A2

Abstract. Probabilistic reasoning in Bayesian networks is normally conducted on a junction tree by repeatedly applying the local propagation whenever new evidence is observed. In this paper, we suggest to treat probabilistic reasoning as database queries. We adapt a method for answering queries in database theory to the setting of probabilistic reasoning in Bayesian networks. We show an effective method for probabilistic reasoning *without* repeated application of local propagation whenever evidence is observed.

1 Introduction

Bayesian networks [3] have been well established as a model for representing and reasoning with uncertain information using probability. Probabilistic reasoning simply means computing the marginal distribution for a set of variables, or the conditional probability distribution for a set of variables given evidence. A Bayesian network is normally transformed through moralization and triangulation into a junction tree on which the probabilistic reasoning is conducted.

One of the most popular methods for performing probabilistic reasoning is the so-called local propagation method [2]. The local propagation method is applied to the junction tree so that the junction tree reaches a consistent state, i.e., a marginal distribution is associated with each node in the junction tree. Probabilistic reasoning can then be subsequently carried out on this consistent junction tree [2].

In this paper, by exploring the intriguing relationship between Bayesian networks and relational databases [5], we propose a new approach for probabilistic reasoning by treating it as a database query. This new approach has several salient features. (1) It advocates using hypertree instead of junction tree for probabilistic reasoning. By selectively pruning the hypertree, probabilistic reasoning can be performed by employing the local propagation method *once* and can then answer any queries without another application of local propagation. (2) The structure of a fixed junction tree may be favourable to some queries but not to others. By using the hypertree as the secondary structure, we can dynamically prune the hypertree to obtain the best choice for answering each query. (3) Finally, this database perspective of probabilistic reasoning provides ample opportunities for well developed techniques in database theory, especially

techniques in query optimization, to be adopted for achieving more efficient probabilistic reasoning in practice.

The paper is organized as follows. We briefly review Bayesian networks and local propagation in Sect. 2. In Sect. 3, we first discuss the relationship between hypertrees and junction trees and the notion of Markov network. We then present our proposed method. We discuss the advantages of the proposed method in Sect. 4. We conclude our paper in Sect. 5.

2 Bayesian Networks and the Local Propagation

We use $U = \{x_1, \dots, x_n\}$ to represent a set of discrete variables. Each x_i takes values from a finite domain denoted V_{x_i} . We use capital letters such as X to represent a subset of U and its domain is denoted by V_X . By XY we mean $X \cup Y$. We write $x_i = \alpha$, where $\alpha \in V_{x_i}$, to indicate that the variable x_i is instantiated to the value α . Similarly, we write $X = \beta$, where $\beta \in V_X$, to indicate that X is instantiated to the value β . For convenience, we write $p(x_i)$ to represent $p(x_i = \alpha)$ for all $\alpha \in V_{x_i}$. Similarly, we write $p(X)$ to represent $p(X = \beta)$ for all $\beta \in V_X$.

A *Bayesian network* (BN) defined over a set $U = \{x_1, \dots, x_n\}$ of variables is a tuple (\mathcal{D}, C) . The \mathcal{D} is a *directed acyclic graph* (DAG) and the $C = \{p(x_i|pa(x_i)) \mid x_i \in U\}$ is a set of *conditional probability distributions* (CPDs), where $pa(x_i)$ denotes the parents of node x_i in \mathcal{D} , such that $p(U) = \prod_{i=1}^n p(x_i|pa(x_i))$. This factorization of $p(U)$ is referred to as *BN factorization*. Probabilistic reasoning in a BN means computing $p(X)$ or $p(X|E = e)$, where $X \cap E = \emptyset$, $X \subseteq U$, and $E \subseteq U$. The fact that E is instantiated to e , i.e., $E = e$, is called the *evidence*.

The DAG of a BN is normally transformed through moralization and triangulation into a junction on which the local propagation method is applied [1]. After the local propagation finishes its execution, a marginal distribution is associated with each node in the junction tree. $p(X)$ can be easily computed by identifying a node in the junction tree which contains X and then do a marginalization; the probability of $p(X|E = e)$ can be similarly obtained by first incorporating the evidence $E = e$ into the junction tree and then propagating the evidence so that the junction tree reaches an updated consistent state from which the probability $p(X|E = e)$ can be computed in the same fashion as we compute $p(X)$. It is worth emphasizing that this method for computing $p(X|E = e)$ needs to apply the local propagation procedure every time when new evidence is observed. That is, it involves a lot of computation to keep the knowledge base consistent. It is also worth mentioning that it is not evident how to compute $p(X)$ and $p(X|E = e)$ when X is not a subset of any node in the junction tree [6]. A more formal technical treatment on the local propagation method can be found in [1].

The problem of probabilistic reasoning, i.e., computing $p(X|E = e)$, can then be equivalently stated as computing $p(X, E)$ for the set $X \cup E$ of variables [6]. Henceforth, probabilistic reasoning means computing the marginal distribution $p(W)$ for any subset W of U .

3 Probabilistic Reasoning as Database Queries

In this section, we show that the task of computing $p(X)$ for any arbitrary subset X can be conveniently expressed and solved as database query.

3.1 Hypertrees, Junction Trees, and Markov Networks

A hypergraph is a pair $(\mathbf{N}, \mathcal{H})$, where \mathbf{N} is a finite set of nodes (attributes) and \mathcal{H} is a set of edges (hyperedges) which are arbitrary subsets of \mathbf{N} [4]. If the nodes are understood, we will use \mathcal{H} to denote the hypergraph $(\mathbf{N}, \mathcal{H})$. We say an element h_i in a hypergraph \mathcal{H} is a *twig* if there exists another element h_j in \mathcal{H} , distinct from h_i , such that $(\cup(\mathcal{H} - \{h_i\})) \cap h_i = h_i \cap h_j$. We call any such h_j a *branch* for the twig h_i . A hypergraph \mathcal{H} is a *hypertree* [4] if its elements can be ordered, say h_1, h_2, \dots, h_n , so that h_i is a twig in $\{h_1, h_2, \dots, h_i\}$, for $i = 2, \dots, n-1$. We call any such ordering a *hypertree (tree) construction ordering* for \mathcal{H} . It is noted for a given hypertree, there may exist multiple tree construction orderings. Given a tree construction ordering h_1, h_2, \dots, h_n , we can choose, for each i from 2 to N , an integer $j(i)$ such that $1 \leq j(i) \leq i-1$ and $h_{j(i)}$ is a branch for h_i in $\{h_1, h_2, \dots, h_i\}$. We call a function $j(i)$ that satisfies this condition a *branching* for the hypertree \mathcal{H} with h_1, h_2, \dots, h_n being the tree construction ordering. For a given tree construction ordering, there might exist multiple choices of branching functions. Given a tree construction ordering h_1, h_2, \dots, h_n for a hypertree \mathcal{H} and a branching function $j(i)$ for this ordering, we can construct the following multiset: $\mathcal{L}(\mathcal{H}) = \{h_{j(2)} \cap h_2, h_{j(3)} \cap h_3, \dots, h_{j(n)} \cap h_n\}$. The multiset $\mathcal{L}(\mathcal{H})$ is the same for any tree construction ordering and branching function of \mathcal{H} [4]. We call $\mathcal{L}(\mathcal{H})$ the *separator set* of the hypertree \mathcal{H} .

Let $(\mathbf{N}, \mathcal{H})$ be a hypergraph. Its *reduction* $(\mathbf{N}, \mathcal{H}')$ is obtained by removing from \mathcal{H} each hyperedge that is a proper subset of another hyperedge. A hypergraph is *reduced* if it equals its reduction. Let $M \subseteq \mathbf{N}$ be a set of nodes of the hypergraph $(\mathbf{N}, \mathcal{H})$. The *set of partial edges* generated by M is defined to be the reduction of the hypergraph $\{h \cap M \mid h \in \mathcal{H}\}$.

Let \mathcal{H} be a hypertree and X be a set of nodes of \mathcal{H} . The set of partial edges generated by X is also a hypertree [7]. A hypergraph \mathcal{H} is a hypertree if and only if its reduction is a hypertree [4]. Henceforth, we will treat each hypergraph as if it is reduced unless otherwise noted.

It has been shown in [4] that given a hypertree, there exists a set of junction trees each of which corresponds to a particular tree construction ordering and a branching function for this ordering. On the other hand, given a junction tree, there always exists a unique corresponding hypertree representation whose hyperedges are the nodes in the junction tree.

Example 1. Consider the hypertree \mathcal{H} shown in Fig 1 (i), it has three corresponding junction trees shown in Fig 1 (ii), (iii) and (iv), respectively. On the other hand, each of the junction trees in Fig 1 (ii), (iii) and (iv) corresponds to the hypertree in Fig 1 (i). The hypertree in Fig 1 (v) will be explained later.

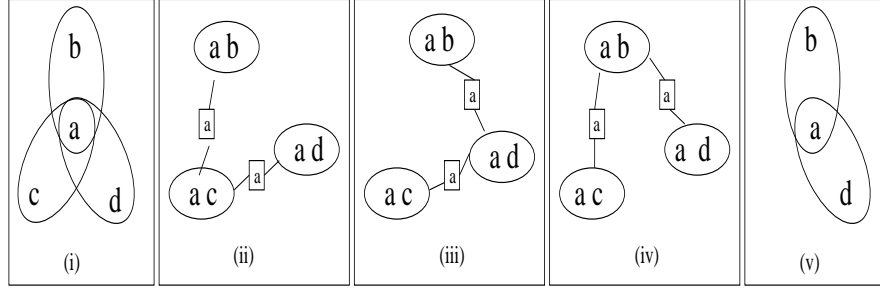


Fig. 1. (i) A hypertree \mathcal{H} , and its three possible junction tree representations in (ii), (iii), and (iv). The pruned hypertree with respect to b, d is in (v).

We now introduce the notion of Markov network. A (decomposable) *Markov network* (MN) [3] is a pair (\mathcal{H}, P) , where (a) $\mathcal{H} = \{h_i | i = 1, \dots, n\}$ is a hypertree defined over variable set U where $U = \bigcup_{h_i \in \mathcal{H}} h_i$ with h_1, \dots, h_n as a tree construction ordering and $j(i)$ as the branching function; together with (b) a set $P = \{p(h) | h \in \mathcal{H}\}$ of *marginals* of $p(U)$. The conditional independencies encoded in \mathcal{H} mean that the jpd $p(U)$ can be expressed as *Markov factorization*:

$$p(U) = \frac{p(h_1) \cdot p(h_2) \cdot \dots \cdot p(h_n)}{p(h_2 \cap h_{j(2)}) \cdot \dots \cdot p(h_n \cap h_{j(n)})}. \quad (1)$$

Recall the local propagation method for BNs in Sect. 2. After finishing the local propagation without any evidence, we have obtained marginals for each node in the junction tree. Since a junction tree corresponds to a unique hypertree, the hypertree and the set of marginals (for each hyperedge) obtained by local propagation together define a Markov network [1].

3.2 An Illustrative Example For Computing $p(X)$

As mentioned before, the probabilistic reasoning can be stated as the problem of computing $p(X)$ where X is an arbitrary set of variables.

Consider a Markov network obtained from a BN by local propagation and let \mathcal{H} be its associated hypertree. It is trivial to compute $p(X)$ if $X \subseteq h$ for some $h \in \mathcal{H}$, as $p(X) = \sum_{h-X} p(h)$. However, it is not evident how to compute $p(X)$ in the case that $X \not\subseteq h$. Using the Markov network obtained by local propagation, we use an example to demonstrate how to compute $p(X)$ for any arbitrary subset $X \subset U$ by selectively pruning the hypertree \mathcal{H} .

Example 2. Consider the Markov network whose associated hypertree \mathcal{H} is shown in Fig 1 (i) and its Markov factorization as follows:

$$p(abcd) = \frac{p(ab) \cdot p(ac) \cdot p(ad)}{p(a) \cdot p(a)}. \quad (2)$$

Suppose we want to compute $p(bd)$ where the nodes b and d are not contained by any hyperedge of \mathcal{H} . We can compute $p(bd)$ by marginalizing out all the

irrelevant variables in the Markov factorization of the jpd $p(abcd)$ in equation (2).

Notice that the numerator $p(ac)$ in equation (2) is the only factor that involves variable c . (Graphically speaking, the node c occurs only in the hyperedge ac). Therefore, we can sum it out as shown below.

$$\begin{aligned} p(bd) &= \sum_{a, c} \frac{p(ab) \cdot p(ac) \cdot p(ad)}{p(a) \cdot p(a)} = \sum_a \frac{p(ab) \cdot p(ad)}{p(a) \cdot p(a)} \sum_c p(ac) \\ &= \sum_a \frac{p(ab) \cdot p(ad)}{p(a)} \cdot \frac{p(a)}{p(a)} = \sum_a \frac{p(ab) \cdot p(ad)}{p(a)}. \end{aligned} \quad (3)$$

Note that $\frac{p(ab) \cdot p(ad)}{p(a)} = p(abd)$ and this is a Markov factorization. The above summation process graphically corresponds to deleting the node c from the hypertree \mathcal{H} in Fig 1 (i), which results in the hypertree in Fig 1 (iv). Note that after the variable c has been summed out, there exists $p(a)$ both as a term in the numerator and a term in the denominator. The existence of the denominator term $p(a)$ is due to the fact that a is in $\mathcal{L}(\mathcal{H})$. Obviously, this pair of $p(a)$ can be canceled. Therefore, our original objective of computing $p(bd)$ can now be achieved by working with this “pruned” Markov factorization $p(abd) = \frac{p(ab) \cdot p(ad)}{p(a)}$ whose hypertree is shown in Fig 1 (iv).

3.3 Selectively Pruning the Hypertree of the Markov Network

The method demonstrated in Example 2 can actually be generalized to compute $p(X)$ for any $X \subset U$. In the following, we introduce a method for selectively pruning the hypertree \mathcal{H} associated with a Markov network to the exact portion needed for computing $p(X)$. This method was originally developed in the database theory for answering database queries [7].

Consider a Markov network with its associated hypertree \mathcal{H} and suppose we want to compute $p(X)$. In order to reduce the hypertree \mathcal{H} to the exact portion that facilitates the computation of $p(X)$, we first mark those nodes in X and repeat the following two operations to prune the hypertree \mathcal{H} until neither is applicable: (op1): delete an unmarked node that occurs in only one hyperedge; (op2): delete a hyperedge that is contained in another hyperedge. We use \mathcal{H}' to denote the resulting hypergraph.

Note that the above procedure possesses the Church-Rosser property, that is, the final result \mathcal{H}' is unique, regardless of the order in which (op1) and (op2) are applied [4]. It is also noted that the operators (op1) and (op2) can be implemented in linear time [7]. It is perhaps worth mentioning that (op1) and (op2) are *graphical* operators applying to \mathcal{H} . On the other hand, the method in [8] works with BN factorization and it sums out irrelevant variables *numerically*.

Let $\mathcal{H}_0, \mathcal{H}_1, \dots, \mathcal{H}_j, \dots, \mathcal{H}_m$ represent the sequence of hypergraphs (not necessarily reduced) in the pruning process, where $\mathcal{H}_0 = \mathcal{H}$, $\mathcal{H}_m = \mathcal{H}'$, $1 \leq j \leq m$, each \mathcal{H}_j is obtained by applying either (op1) or (op2) to \mathcal{H}_{j-1} .

Lemma 1. Each hypergraph \mathcal{H}_i , $1 \leq i \leq m$, is a hypertree.

Lemma 2. $\mathcal{L}(\mathcal{H}') \subseteq \mathcal{L}(\mathcal{H})$.

Due to lack of space, the detailed proofs of the above lemmas and the following theorem will be reported in a separate paper.

For each $h'_i \in \mathcal{H}'$, there exists a $h_j \in \mathcal{H}$ such that $h'_i \subseteq h_j$. We can compute $p(h'_i)$ as $p(h'_i) = \sum_{h_j \supseteq h'_i} p(h_j)$. In other words, the marginal $p(h'_i)$ can be computed from the original marginal $p(h)$ supplied with the Markov network \mathcal{H} . Therefore, after selectively pruning \mathcal{H} , we have obtained the hypertree \mathcal{H}' and marginals $p(h'_i)$ for each $h'_i \in \mathcal{H}'$.

Theorem 1. Let (\mathcal{H}, P) be a Markov network. Let \mathcal{H}' be the resulting pruned hypertree with respect to a set X of variables. The hypertree $\mathcal{H}' = \{h'_1, h'_2, \dots, h'_k\}$ and the set $P' = \{p(h'_i) \mid 1 \leq i \leq k\}$ of marginals define a Markov network.

Theorem 1 indicates that the original problem of computing $p(X)$ can now be answered by the new Markov network defined by the pruned hypertree \mathcal{H}' . It has been proposed [5] that each marginal $p(h)$ where $h \in \mathcal{H}$ can be stored as a relation in the database fashion. Moreover, computing $p(X)$ from \mathcal{H}' can be implemented by database SQL statements [5]. It is noted that the result of applying (op1) and (op2) to \mathcal{H} always yields a Markov network which is different than the method in [8].

4 Advantages

One salient feature of the proposed method is that it does not require any repeated application of local propagation. Since the problem of computing $p(X|E=e)$ can be equivalently reduced to the problem of computing $p(X, E)$, we can uniformly treat probabilistic reasoning as merely computing marginal distributions. Moreover, computing a marginal distribution, say, $p(W)$, from the jpd $p(U)$ defined by a BN, can be accomplished by working with the Markov factorization of $p(U)$, whose establishment only needs applying the local propagation method once on the junction tree constructed from the BN. It is worth mentioning that Xu in [6] reached the same conclusion and proved a theorem similar to Theorem 1 based on the local propagation technique on the junction tree.

Computing $p(W)$ in our proposed approach needs to prune the hypertree \mathcal{H} to the exact portion needed for computing $p(W)$ as Example 2 demonstrates if $W \not\subseteq h$ for any $h \in \mathcal{H}$. A similar method was suggested in [6] by pruning the junction tree instead. Using hypertrees as the secondary structure instead of junction trees has valuable advantages as the following example shows.

Example 3. Consider the hypertree \mathcal{H} shown in Fig 1 (i), it has three corresponding junction trees shown in Fig 1 (ii), (iii) and (iv), respectively. The method in [6] first fixes a junction tree, for example, say the one in Fig 1 (ii). Suppose we need to compute $p(bd)$, the method in [6] will prune the junction tree so that any irrelevant nodes will be removed as we do for pruning hypertree. However, in this

junction tree, nothing can be pruned out according to [6]. In other words, $p(bd)$ has to be obtained by the following calculation: $p(bd) = \sum_a \frac{p(ab)}{p(a)} \cdot \sum_c \frac{p(ac) \cdot p(ad)}{p(a)}$. However, if we prune the hypertree in Fig 1 (i), the resulting pruned hypertree is shown in Fig 1 (v), from which $p(bd)$ can be obtained by equation (3). Obviously, the computation involved is much less. Observing this, one might decide to adopt the junction tree in Fig 1 (iii) as the secondary structure. This change facilitates the computation of $p(bd)$. However, in a similar fashion, one can easily be convinced that computing $p(bc)$ using the junction tree in (iii), computing $p(cd)$ using the junction tree in (iv) suffer exactly the same problem as computing $p(bd)$ using junction tree in (ii). In other words, regardless of the junction tree fixed in advance, there always exists some queries that are not favored by the pre-determined junction tree structure. On the other hand, the hypertree structure always provides the optimal pruning result for computing marginal [7].

5 Conclusion

In this paper, we have suggested a new approach for conducting probabilistic reasoning from the relational database perspective. We demonstrated how to selectively reduce the hypertree structure so that we can avoid repeated application of local propagation. This suggests a possible dual purposes database management systems for both database storage, retrieval and probabilistic reasoning.

ACKNOWLEDGMENT. The authors would like to thank one of the reviewers for his/her helpful suggestions and comments.

References

- [1] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, October 1996.
- [2] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50:157–244, 1988.
- [3] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, California, 1988.
- [4] G. Shafer. An axiomatic study of computation in hypertrees. School of Business Working Papers 232, University of Kansas, 1991.
- [5] S.K.M. Wong, C.J. Butz, and Y. Xiang. A method for implementing a probabilistic model as a relational database. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 556–564. Morgan Kaufmann Publishers, 1995.
- [6] H. Xu. Computing marginals for arbitrary subsets from marginal representation in markov trees. *Artificial Intelligence*, 74:177–189, 1995.
- [7] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 82–94, 1981.
- [8] N. Zhang and Poole.D. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.