A Method for Constructing the Dependency Structure of a Probabilistic Network

S.K.M. Wong
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
wong@cs.uregina.ca

C.J. Butz
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
butz@cs.uregina.ca

Abstract

Probabilistic reasoning has become an accepted formulism for managing uncertainty in Artificial Intelligence. The usual input to a probabilistic model is a Bayesian network containing both embedded and nonembedded conditional independence information. A Bayesian network comprises a qualitative and a quantitative component, namely a directed acyclic graph explicitly specifying the dependency structure of the network, coupled with a set of corresponding conditional probability tables. To reduce the computational complexity of probabilistic inference it is useful to transform a Bayesian network into a Markov network albeit sacrificing the embedded conditional independency information. Another method for constructing the dependency structure of a Markov network is to apply a learning algorithm to a repository of observed data. In this paper, a method is suggested for constructing the structure of a Markov network from a given input set of nonembedded conditional independencies. This technique involves determining a conflict-free cover of such independencies. A unique Markov structure can then be systematically constructed from the cover. The proposed approach takes full advantage of a recent result that nonembedded conditional independencies have a *complete* axiomatization. This result provides a basis for constructing a dependency structure which is a perfect map of the input set of nonembedded conditional independencies. That is, every nonembedded conditional independence logically implied by the input set can be inferred from the dependency structure, and every nonembedded conditional independence inferred from the dependency structure is logically implied by the input set. The problem of constructing a Bayesian network given an input set of *both* embedded and nonembedded conditional independencies is currently being investigated.

1 Introduction

Probabilistic reasoning has become an accepted formulism for managing uncertainty in Artificial Intelligence. The usual input to the probabilistic model of a Bayesian network [11, 14, 15] contains both *embedded* and *nonembedded* conditional independence information. A Bayesian network comprises a qualitative and a quantitative component, namely a directed acyclic graph explicitly specifying the dependency structure of the network, coupled with a set of corresponding conditional probability tables. Such a network utilizes conditional independencies to provide an equivalent economical representation of a joint probability distribution. To reduce the computational complexity of probabilistic inference, it is useful to transform a Bayesian network into a Markov network [9] albeit sacrificing the embedded conditional independency information.

One approach to constructing a probabilistic network is to first learn the qualitative component [10, 15, 19, 21, 23] and then elicit the quantitative component [6] from the domain expert. The techniques for learning the qualitative component can be further classified into those which learn embedded conditional independencies and those which learn nonembedded conditional independencies. For nonembedded techniques [21, 23], the dependency structure of a Markov network is learned from observed data. This technique can be extended recursively to learn embedded conditional independencies [10]. It is known that the learned dependency structure is not necessarily a *perfect map* [15] as it has been shown that discovering all the probabilistic conditional independencies in ob-

served data is a NP-hard problem [5]. Alternatively, the dependency structure of a probabilistic network can be constructed from an input set of conditional independencies. Verma and Pearl [19] derived conditions of equivalence and synthesis of dependency structures containing both embedded and nonembedded conditional information from an input set of conditional independencies. However, it has been proved that there is no complete axiomatization of embedded conditional independencies using a finite set of inference rules [18, 22], contrary to Pearl's [15] conjecture. Thus, even though a dependency structure can be constructed which represents the embedded conditional independencies in the input set, one cannot be sure that the constructed dependency structure is a perfect map of the input set. That is, an embedded conditional independence logically implied by the input set may not be inferred from the dependency structure. Recently, however, it has been shown that nonembedded conditional independencies do indeed have a *complete* axiomatization [20]. This result provides us with a tool for constructing a dependency structure which is a perfect map of the input set of nonembedded conditional independencies. That is, every nonembedded conditional independence logically implied by the input set can be inferred from the dependency structure, and every nonembedded conditional independence inferred from the dependency structure is logically implied by the input set. As in the first approach, once the dependency structure has been constructed, whether embedded or nonembedded, the quantitative component of the network is then elicited from the domain expert [6].

In this paper we suggest a method for constructing the dependency structure of a Markov network from an input set of conditional independencies. A cover representing exactly the probabilistic conditional independencies in the input set is constructed. All redundant conditional independencies are removed deriving a minimum cover in the process of enforcing conflict freedom [3]. A conflict-free cover has the desirable property that every conditional independence in the cover is used in the construction of the network structure. In other words, the situation in which the use of one conditional independency excludes the use of another in the construction of the network does not happen. Thus, the conditional independencies in a conflict-free cover can be systematically applied to construct a unique dependency structure. The discussion in this paper can be seen as an extension of designing a database schema in the relational database theory given a set of multivalued dependencies [3] into a probabilistic framework.

This paper is organized as follows. Section 2 contains background knowledge. In Section 3, we describe the proposed method for constructing the dependency structure of a probabilistic network from an input set of conditional independencies, and show that it is a perfect map. The conclusion is presented in Section 4.

2 Background

2.1 Hypergraphs and Hypertrees

Let N be a finite set of variables $\{A_1, A_2, \ldots, A_m\}$. A *hypergraph*, denoted $\mathcal{H}$, is a family of subsets of variables in N, i.e., $\mathcal{H} \subseteq 2^N$. An element in $\mathcal{H}$ is called a *hyperedge*.

We call an element $t \in \mathcal{H}$, a *twig*, if there exists another distinct element $b \in \mathcal{H}$, such that $t \cap (\cup(\mathcal{H} - \{t\})) = t \cap b$. (By this definition, the hyperedge in a hypergraph consisting of a single hyperedge is not a twig). This means that the intersection of t and the hypergraph is contained in one hyperedge of the hypergraph. We call any such b a *branch* for the twig t, and note that a twig t may have many possible branches. A hypergraph is called a *hypertree* (an acyclic hypergraph [3, 17]) if its elements can be ordered, $h_1, h_2, \ldots, h_i$, such that h_i is a twig in the sub-hypergraph $h_1, h_2, \ldots, h_i$ for $i = 1, \ldots, n$. We call any ordering satisfying this condition a hypertree *construction ordering* for $\mathcal{H}$. (A hypertree construction ordering can also be represented as a *join tree* [3].) The first hyperedge h_1 in the hypertree construction ordering is called the root. Given a particular hypertree construction ordering, we can choose an integer $b(i)$, for $i = 2, \ldots, n$, such that $1 \leq b(i) \leq i - 1$ and $h_{b(i)}$ is a branch for h_i in $h_1, h_2, \ldots, h_i$. We call such a function $b(i)$ satisfying this condition a *branching function* for $\mathcal{H}$. Note that a particular construction ordering may have many branching functions.

Example 1
Consider the case where $N = \{A_1, A_2, \ldots, A_6\}$. Let $\mathcal{H} = \{h_1 = \{A_1, A_2, A_3\}, h_2 = \{A_1, A_2, A_4\}, h_3 = \{A_1, A_2, A_5\}, h_4 = \{A_5, A_6\}\}$ denote the hypergraph shown in Figure 1. Since we can define a hypertree construction ordering h_1, h_2, h_3, h_4, this hypergraph is a hypertree. One possible branching function for this hypertree construction ordering h_1, h_2, h_3, h_4 is $b(2) = 1, b(3) = 1, b(4) = 3$.

Given a hypertree construction ordering $h_1, h_2, \ldots, h_n$ for a hypertree $\mathcal{H}$, and a branching function $b(i)$ for this ordering, we can construct the following set of subsets: $\mathcal{L} = \{h_{b(2)} \cap h_2, h_{b(3)} \cap h_3, \ldots, h_{b(n)} \cap h_n\} = \{l_2, \ldots, l_n\}$. This set $\mathcal{L}$ is in fact independent of the hypertree construction ordering, i.e., $\mathcal{L}$ is the same for any tree construction ordering of a given hypertree.

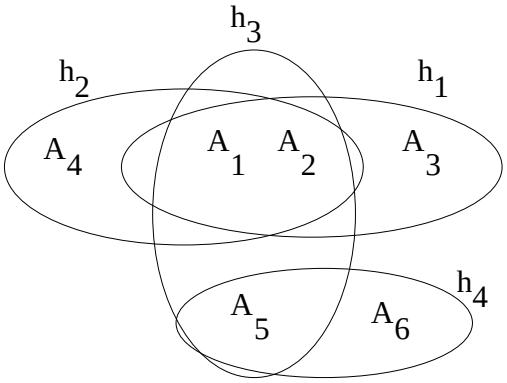

Figure 1: A graphical representation of the hypergraph $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$.

We call $\mathcal{L}$ the set of *J-keys* of the hypertree $\mathcal{H}$. ($\mathcal{L}$ is also called a *d-sepset* or *intersection set* in the literature.)

Let $\mathcal{H}$ be a hypergraph. The graph of $\mathcal{H}$, denoted $G(\mathcal{H})$, has the same nodes as $\mathcal{H}$ and an edge between every pair of nodes that are in the same hyperedge of $\mathcal{H}$. Hence, the edges of $G(\mathcal{H})$ are precisely the set of all pairs (A_i, A_j) for which there is an hyperedge $h \in \mathcal{H}$ such that $A_i, A_j \in h$.

A hypergraph is *conformal* [4] if for every clique V in $G(\mathcal{H})$ there is a hyperedge of $\mathcal{H}$ that contains V. It has been shown [3] that $\mathcal{H}$ is an acyclic hypergraph if and only if $\mathcal{H}$ is a chordal conformal hypergraph. As well, if $\mathcal{H}$ is chordal then $G(\mathcal{H})$ is chordal (triangulated).

2.2 Factored Probability Distributions

A joint probability distribution $\phi(A_1, A_2, \ldots, A_m)$ can be factorized into a product of several terms, called *potentials*, as:

$$\phi(A_1, A_2, \ldots, A_m) = \phi_{h_1} \cdot \phi_{h_2} \cdot \ldots \cdot \phi_{h_n},$$

where h_i is a non-empty subset of variables in $N = \{A_1, A_2, \ldots, A_m\}$, $1 \leq i \leq m$, and ϕ_{h_i} is a nonnegative real valued function on h_i. Thus, a joint probability distribution can be written as:

$$\phi(A_1, A_2, \ldots, A_m) = \prod_{h \in \mathcal{H}} \phi_h, \qquad (1)$$

where $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ is a hypergraph. (Here we assume that $\cup\mathcal{H} = \cup_{i=1}^{n} h_i = N$. In general, some hyperedge $h_i \in \mathcal{H}$ may be a proper subset of another hyperedge h_j in $\mathcal{H}$, i.e., $h_i \subset h_j$.)

2.3 Bayesian and Markov Distributions

Let $N = \{A_1, A_2, \ldots, A_m\}$ be a set of vertices (variables), and E be a set of ordered pairs (A_i, A_j), i.e., $(A_i, A_j) \in N \times N$. The ordered pair (A_i, A_j) reflects a directed edge from A_i to A_j. Let $D = (N, E)$ be a directed acyclic graph (DAG). The arcs in D reflect the dependencies that hold amongst the variables. A Bayesian network [11, 14, 15] is a pair (D, P) where P is a set of conditional probabilities. The joint probability distribution ϕ can be written as:

$$\phi(A_1, A_2, \ldots, A_m) = \prod_{i=1}^{m} \phi(A_i \mid pa(A_i)), \qquad (2)$$

where $\phi(A_i \mid pa(A_i)) \in P$, and $pa(A_i)$ is the parent set of A_i defined as $pa(A_i) = \{A_j \mid (A_j, A_i) \in E\}$. We refer to ϕ in Equation (2) as a Bayesian distribution.

Example 2 The DAG in Figure 2 depicts the factorization of the joint probability distribution ϕ, namely:

$$\begin{aligned} \phi(A_1, A_2, \ldots, A_5) \quad = \quad & \phi(A_1) \cdot \phi(A_2 \mid A_1) \cdot \phi(A_3 \mid A_1) \cdot \\ & \phi(A_4 \mid A_2, A_3) \cdot \phi(A_5 \mid A_4). \end{aligned} \qquad (3)$$

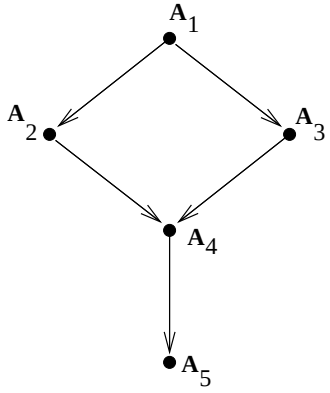

Figure 2: The directed acyclic graph (DAG) of the Bayesian network defined by the factorized distribution of Equation (3).

Given the DAG of a Bayesian network, we know precisely what conditional probabilities are required to construct the joint distribution. Thus, conditional independence assumptions provide a practical way to define a joint distribution, particularly in a situation where the number of variables involved is very large. However, in practice, it may still take considerable amount of time and space to compute the required marginals of a factorized distribution. For this reason, many efficient algorithms based on local computations have been developed for computing the marginals of a Markov networks [9]. (Note that this definition of Markov network is different from that given in [15].)

A joint probability distribution ϕ on $N = \{A_1, A_2, \ldots, A_m\}$ is called a Markov distribution, if

ϕ can be factorized on a hypertree $\mathcal{H}$ as:

$$\phi(A_1, A_2, \ldots, A_m) = \frac{\prod_{h \in \mathcal{H}} \phi^{\downarrow h}}{\prod_{l \in \mathcal{L}} \phi^{\downarrow l}}, \quad (4)$$

where $\mathcal{L}$ is the set of J-keys of $\mathcal{H}$, and $\phi^{\downarrow h}$ denotes the marginal distribution of ϕ onto the subset of variables h [9, 14].

For example, the decomposition of the following distribution:

$$\begin{aligned} & \phi(A_1, A_2, A_3, A_4, A_5) \\ = \; & \frac{\phi^{\downarrow\{A_1, A_2, A_3\}} \cdot \phi^{\downarrow\{A_2, A_3, A_4\}} \cdot \phi^{\downarrow\{A_4, A_5\}}}{\phi^{\downarrow\{A_2, A_3\}} \cdot \phi^{\downarrow\{A_4\}}}, \end{aligned} \quad (5)$$

can be conveniently represented by a chordal undirected graph depicting the dependency structure of the Markov distribution as shown in Figure 3.

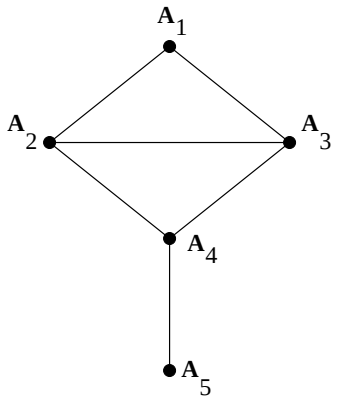

Figure 3: The chordal undirected graph depicting the dependency structure of the Markov distribution defined by equation (5).

It is evident that in this context Markov distributions are less expressive than Bayesian distributions. For instance, the conditional independence $\phi(A_2, A_3|A_1) = \phi(A_2|A_1) \cdot \phi(A_3|A_1)$, embedded in Equation (3) is lost in Equation (5). An *embedded* conditional independence is one which does not hold with respect to the joint probability distribution defined on a set of attributes N, but does hold with respect to some marginal distribution defined on a proper subset of attributes $N' \subset N$. As the concept of nonembedded conditional independence is important for our discussion, we distinguish it from an embedded conditional independence with the aid of an example.

Consider the joint probability distribution ϕ on $N = \{A_1, A_2, A_3, A_4\}$ as shown in Figure 4.

It can be verified that the conditional independence of $\{A_2\}$ and $\{A_3\}$ given $\{A_1\}$ does not hold with respect to $\phi(N)$. That is,

$$\phi(A_1, A_2, A_3, A_4) \neq \frac{\phi^{\downarrow\{A_1, A_2\}} \cdot \phi^{\downarrow\{A_1, A_3, A_4\}}}{\phi^{\downarrow\{A_1\}}}.$$

$\phi(A_1, A_2, A_3, A_4) =$

A_1	A_2	A_3	A_4	f_ϕ
0	0	0	0	0.2
0	0	1	1	0.2
0	1	0	0	0.2
0	1	1	0	0.2
1	1	1	1	0.2

Figure 4: The joint probability distribution ϕ on $N = \{A_1, A_2, A_3, A_4\}$.

However, the conditional independence of $\{A_2\}$ and $\{A_3\}$ given $\{A_1\}$ holds in the marginal distribution $\phi^{\downarrow\{A_1, A_2, A_3\}}$, namely:

$$\begin{aligned} & \phi^{\downarrow\{A_1, A_2, A_3\}} \\ = \; & \frac{(\phi^{\downarrow\{A_1, A_2, A_3\}})^{\downarrow\{A_1, A_2\}} \cdot (\phi^{\downarrow\{A_1, A_2, A_3\}})^{\downarrow\{A_1, A_3\}}}{(\phi^{\downarrow\{A_1, A_2, A_3\}})^{\downarrow\{A_1\}}}. \end{aligned}$$

We call such an independency an embedded conditional independency with respect to the distribution $\phi(A_1, A_2, A_3, A_4)$.

Even without embedded conditional independencies, it has been amply demonstrated that Markov distributions play an important role in the design of efficient algorithms for probabilistic reasoning.

3 Construction of the Dependency Structure

In this section we demonstrate a technique for constructing the dependency structure of a Markov distribution. The input to our procedure is a finite set of conditional independencies. The first task is to compute a logically equivalent cover which succinctly expresses the dependency bases of the input set of conditional independencies. All redundant conditional independencies are removed from the cover in the process of enforcing conflict freedom. The resulting conflict-free cover is the minimum cover of the input set. Finally, an algorithm to construct a unique dependency structure from the conflict-free cover is presented.

It has been proved [18, 22] that there is no complete axiomatization of embedded conditional independencies using a finite set of inference rules, contrary to Pearl's conjecture [15]. However, it has recently been shown that nonembedded conditional independencies do indeed have a complete axiomatization [20]. This result provides us with a tool for constructing a dependency structure which is a *perfect map* of the input

set of nonembedded conditional independencies. That is, every nonembedded conditional independence logically implied by the input set can be inferred from the derived dependency structure, and every nonembedded conditional independence inferred from the dependency structure is logically implied by the input set.

It should be noted that nonembedded conditional independence is termed *generalized multivalued dependency* (GMVD) in the extended relational model [20].

Definition 1 [20] Let $X, Y, Z \subseteq N$. We say that a distribution ϕ on N satisfies the generalized multivalued dependency (GMVD) $X \multimap\!\!\rightarrow Y$ if

$$\phi(XYZ) = \frac{\phi^{\downarrow XY} \cdot \phi^{\downarrow XZ}}{\phi^{\downarrow X}},$$

where $Z = N - XY$.

In the following discussion, the nonembedded conditional independence of Y and Z given X, will henceforth be referred to as the GMVDs $X \multimap\!\!\rightarrow Y$ and $X \multimap\!\!\rightarrow Z$. The *complete minimal* set of inference rules for GMVDs [20] are: for all subsets $X, Y, Z, W \subseteq N$,

M1: If $X \multimap\!\!\rightarrow Y$, then $X \multimap\!\!\rightarrow N - XY$,
M2: If $Y \subseteq X$, then $X \multimap\!\!\rightarrow Y$,
M3: If $Z \subseteq W$ and $X \multimap\!\!\rightarrow Y$, then $WX \multimap\!\!\rightarrow ZY$,
M4: If $X \multimap\!\!\rightarrow Y$ and $Y \multimap\!\!\rightarrow Z$, then $X \multimap\!\!\rightarrow Z - Y$.

The inference rules $(M1), (M2), (M3), (M4)$ are called complementation, reflexivity, augmentation and transitivity, respectively. From this minimal set, one can derive additional rules that are particularly useful in the completeness proof, namely:

M5: If $X \multimap\!\!\rightarrow Y$ and $X \multimap\!\!\rightarrow Z$, then $X \multimap\!\!\rightarrow YZ$.
M6: If $X \multimap\!\!\rightarrow Y$ and $X \multimap\!\!\rightarrow Z$, then
$X \multimap\!\!\rightarrow Y \cap Z$, $X \multimap\!\!\rightarrow Y - Z$, and $X \multimap\!\!\rightarrow Z - Y$.

The inference rules $(M5)$ and $(M6)$ are called union and decomposition, respectively. The usefulness of a complete axiomatization lies in the ability to determine the equivalence of two sets of GMVDs. As a consequence, given a set of GMVDs it is possible to determine a logically equivalent minimum cover. Working with a minimum cover compared to the input set is more efficient as many algorithms have an order in the size of the input.

Given a set of GMVDs on N, similar to the relational database theory [2, 7], the subset of GMVDs having the same *left side* X can be expressed in terms of a *dependency basis* $DEP(X) = \{W_1, W_2, \ldots, W_m\}$, namely:

$$X \multimap\!\!\rightarrow W_1 \mid W_2 \mid \ldots \mid W_m, \qquad (6)$$

where $X \cap W_i = \emptyset$ for $i = 1, 2, \ldots, m$ and $\{W_1, W_2, \ldots, W_m\}$ forms a partition of $N - X$.

We now present an algorithm to construct the dependency basis for GMVDs. Beeri [1] was the first to propose a polynomial time algorithm to compute the dependency basis for a set of multivalued dependencies in databases. (Faster algorithms have since been proposed [16].) Here we extend Beeri's algorithm by replacing the complete axiomatization for multivalued dependencies with the complete axiomatization for GMVDs [20].

Algorithm 1 [1]

```
procedure DEP-BASIS(X)
BASIS = {{A} | A ∈ {X}} ∪ {N − X}
change = 1
% Compute closure under (M4)
while change do
{
   change = 0
   for each GMVD W −o→ Z in G
   {
      Y = ∪ {R | R ∈ BASIS and R ∩ W ≠ ∅}
      Z' = Z − Y
      if Z' ≠ ∅ and
      is not equal to a union of elements of BASIS
      {
         change = 1
         BASIS = basis of collection of sets from
         BASIS ∪ {Z'} under (M5),(M6)
      }
   }
}
return (BASIS)
end DEP-BASIS
```

Example 3 Consider a set of GMVDs $\{\{A_5\} \multimap\!\!\rightarrow \{A_2\}, \{A_1, A_5\} \multimap\!\!\rightarrow \{A_3\}\}$ on the set of variables $N = \{A_1, A_2, A_3, A_4, A_5\}$. Then $DEP(\{A_5\}) = \{\{A_2\}, \{A_1, A_3, A_4\}\}$ and $DEP(\{A_1, A_5\}) = \{\{A_2\}, \{A_3\}, \{A_4\}\}$.

Given an input set M of GMVDs, let $\mathbf{X}$ denote the family of left sides of the GMVDs in M; we call each $X \in \mathbf{X}$ a key. A logically equivalent *cover* of M can be determined by applying Algorithm 1 to each key in $\mathbf{X}$. That is, for each key $X \in \mathbf{X}$, all redundancy on the right side of the GMVDs $X \multimap\!\!\rightarrow Y_1, \ldots, X \multimap\!\!\rightarrow Y_n$ is removed in computing the dependency basis $X \multimap\!\!\rightarrow W_1 \mid W_2 \mid \ldots \mid W_m$. In order to construct the *minimum cover* of the input set, *conflicting* GMVDs have to be removed.

The notion of conflict-free multivalued dependencies

was originally introduced by Lien [13] in the study of the relationship between various database models. Here we extend this notion to GMVDs. Following Lee's argument [12], we show [20] that a modified Lien's decomposition algorithm generates a unique dependency structure (an acyclic hypergraph) from a set of conflict-free GMVDs.

Let $X \in \mathbf{X}$ and its dependency basis be defined by equation (6). Suppose $Y \not\subseteq X$ is another key. We say Y is not *split* by X if $Y \subseteq W_i X$ for some $i = 1, 2, \ldots, m$. We say $\mathbf{X}$ is conflict-free if

(i) For any $X, Y \in \mathbf{X}$ and $Y \not\subseteq X$, Y is not split by X, and

(ii) $DEP(X) \cap DEP(Y) \subseteq DEP(X \cap Y)$.

For example, if we add the (redundant) GMVD $\{A_1, A_2, A_5\} \multimap\!\!\rightarrow \{A_3\}$ to the set of GMVDs in Example 3, it is no longer conflict-free since the key $\{A_1, A_2, A_5\}$ is split by the key $\{A_5\}$.

The process of ensuring each pair in the set of keys $\mathbf{X}$ are conflict-free removes all redundancy on the left side of the keys. If the set $\mathbf{X}$ of keys in the remaining cover is still not conflict-free, then the conflicting GMVDs can be revised by the domain expert, and the process repeated. Henceforth, we assume that the conflict-free cover has been constructed for the input set of GMVDs. A conflict-free cover is a minimum cover of the input set of GMVDs.

Let $\mathbf{X}$ denote the set of keys in the conflict-free cover. The keys in $\mathbf{X}$ can be arranged in a sequence $(X_1, X_2, \ldots, X_p)$ called a *p-ordering sequence* such that $X_i \subseteq X_j$ implies $i \leq j$. The decomposition algorithm for GMVDs can now be described.

Algorithm 2

Input: the conflict-free cover of the input set of GMVDs
over a set of attributes N,
a p-ordering sequence $(X_1, X_2, \ldots, X_p)$ of the keys in the conflict-free cover.

$\mathcal{H}^0 := \{N\}$;
for $i := 1$ to p
{
  while $X_i \subseteq h_j$ and $h_j \in \mathcal{H}^{i-1}$
  {
    $\mathcal{H}^i := \mathcal{H}^{i-1} - \{h_j\}$;
    $\mathcal{H}^i := \mathcal{H}^i \cup \{X_i \cup (h_j \cap W) \mid W \in DEP(X_i), h_j \cap W \neq \emptyset\}$;
  }
}
Output: an acyclic hypergraph.

Example 4
Let $N = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}$ and let $(\{A_1\}, \{A_2\}, \{A_3\}, \{A_1, A_2\}, \{A_1, A_3\}, \{A_2, A_3\})$ be a p-ordering sequence of the keys in the conflict-free cover of the following GMVDs:

$$
\begin{aligned}
\{A_1\} &\multimap\!\!\rightarrow \{A_7\} \mid \{A_2, A_3, A_4, A_5, A_6, A_8, A_9\},\\
\{A_2\} &\multimap\!\!\rightarrow \{A_8\} \mid \{A_1, A_3, A_4, A_5, A_6, A_7, A_9\},\\
\{A_3\} &\multimap\!\!\rightarrow \{A_9\} \mid \{A_1, A_2, A_4, A_5, A_6, A_7, A_8\},\\
\{A_1, A_2\} &\multimap\!\!\rightarrow \{A_4\} \mid \{A_7\} \mid \{A_8\} \mid \{A_3, A_5, A_6, A_9\},\\
\{A_1, A_3\} &\multimap\!\!\rightarrow \{A_5\} \mid \{A_7\} \mid \{A_9\} \mid \{A_2, A_4, A_6, A_8\},\\
\{A_2, A_3\} &\multimap\!\!\rightarrow \{A_6\} \mid \{A_8\} \mid \{A_9\} \mid \{A_1, A_4, A_5, A_7\}.
\end{aligned}
$$

The intermediate schemas generated by the GMVD decomposition algorithm are:

$$
\begin{aligned}
\mathcal{H}^1 &= \{\{A_1, A_7\}, \{A_1, A_2, A_3, A_4, A_5, A_6, A_8, A_9\}\},\\
\mathcal{H}^2 &= \{\{A_1, A_7\}, \{A_2, A_8\},\\
&\quad \{A_1, A_2, A_3, A_4, A_5, A_6, A_9\}\},\\
\mathcal{H}^3 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\},\\
&\quad \{A_1, A_2, A_3, A_4, A_5, A_6\}\},\\
\mathcal{H}^4 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\},\\
&\quad \{A_1, A_2, A_3, A_5, A_6\}\},\\
\mathcal{H}^5 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\},\\
&\quad \{A_1, A_3, A_5\}, \{A_1, A_2, A_3, A_6\}\},\\
\mathcal{H}^6 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\},\\
&\quad \{A_1, A_3, A_5\}, \{A_2, A_3, A_6\}, \{A_1, A_2, A_3\}\}.
\end{aligned}
$$

It can be easily verified that the output $\mathcal{H} = \mathcal{H}^6$ is an acyclic hypergraph. A hypertree construction ordering of this schema is $h_1 = \{A_1, A_2, A_3\}$, $h_2 = \{A_1, A_2, A_4\}$, $h_3 = \{A_1, A_3, A_5\}$, $h_4 = \{A_2, A_3, A_6\}$, $h_5 = \{A_1, A_7\}$, $h_6 = \{A_2 A_8\}$, $h_7 = \{A_3 A_9\}$. Thus, the J-keys of $\mathcal{H}$ are:

$$
\begin{aligned}
h_2 \cap h_{b(2)} &= \{A_1, A_2\}, \quad b(2) = 1,\\
h_3 \cap h_{b(3)} &= \{A_1, A_3\}, \quad b(3) = 1,\\
h_4 \cap h_{b(4)} &= \{A_2, A_3\}, \quad b(4) = 1,\\
h_5 \cap h_{b(5)} &= \{A_1\}, \quad b(5) = 2,\\
h_6 \cap h_{b(6)} &= \{A_2\}, \quad b(6) = 4,\\
h_7 \cap h_{b(7)} &= \{A_3\}, \quad b(7) = 3.
\end{aligned}
$$

Theorem 1 Let $\mathcal{H}$ be the constructed dependency structure by applying Algorithm 2 to a conflict-free cover of GMVDs defined on a set N of attributes. Let $\mathbf{X}$ be the set of keys in the conflict-free cover. Then the set $\mathcal{L}$ of J-keys of $\mathcal{H}$ is precisely the set of keys in $\mathbf{X}$.

Proof: We will prove this by induction on n in Algorithm 2. Let the p-ordering sequence of the keys in $\mathbf{X}$ be $(X_1, X_2, \ldots, X_p)$. Basic step $(n = 1)$: The key X_1 and $DEP(X_1) = \{Y_{1_1}, Y_{1_2}, \ldots, Y_{1_m}\}$

are used to construct the acyclic hypergraph $\mathcal{H}^1 = \{h_{1_1}, h_{1_2}, \ldots, h_{1_m}\}$, from $\mathcal{H}^0 = \{N\}$, where $h_{1_i} = \{Y_{1_i} X_1\}$, $1 \leq i \leq m$ and m is the number of disjoint sets in $DEP(X_1)$. This means that $h_{1_i} \cap h_{1_j} = X_1$ for all $h_{1_i}, h_{1_j} \in \mathcal{H}^1$. Thus, X_1 is the only J-key in $\mathcal{H}^1$, namely, $\mathcal{L}^1 = \{X_1\}$. Inductive step: Assume that the set of J-keys of the acyclic hypergraph $\mathcal{H}^k$ is $\mathcal{L}^k = \{X_1, X_2, \ldots, X_k\}$. There exists only one hyperedge in $\mathcal{H}^k$, say h_j, such that $X_{k+1} \subset h_j$; otherwise there is some J-key $X_i \in \{X_1, X_2, \ldots, X_k\}$ that splits X_{k+1} contradicting the assumption that the input cover is conflict-free. According to Algorithm 2, the key X_{k+1} and $DEP(X_{k+1})$ are used to construct the new hyperedges $h_{k+1_1}, h_{k+1_2}, \ldots, h_{k+1_l}$. These hyperedges are added to $\mathcal{H}^k - \{h_j\}$ to form $\mathcal{H}^{k+1}$. Since $h_j = \cup_{i=1}^{l} h_{k+1_i}$, the J-keys $\{X_1, X_2, \ldots, X_k\}$ of $\mathcal{H}^k$ remain J-keys of $\mathcal{H}^{k+1}$. By construction in Algorithm 2, $X_{k+1} \subset h_{k+1_i}$. Since each attribute $A \in h_j$ $(A \notin X_{k+1})$ appears in a unique set in $DEP(X_{k+1})$, it appears in a unique h_{k+1_i}. Thus, $\cap_{i=1}^{l} h_{k+1_i} = X_{k+1}$. Thereby, the only J-key implied by replacing h_j with $h_{k+1_1}, h_{k+1_2}, \ldots, h_{k+1_l}$ is X_{k+1}. Thus, the set of J-keys in $\mathcal{H}^{k+1}$ is $\mathcal{L}^{k+1} = \{X_1, X_2, \ldots, X_k, X_{k+1}\}$. □

Theorem 1 indicates that the set $\mathcal{L}$ of J-keys in the constructed dependency structure $\mathcal{H}$ is precisely the set $\mathbf{X}$ of keys in the conflict-free cover derived from the input set of GMVDs.

In order to show that the constructed dependency structure $\mathcal{H}$ is a perfect map of the input set of GMVDs, it must be demonstrated that the set of all GMVDs inferred from the dependency structure is logically equivalent to the input set of GMVDs.

We can adopt the method in [8] to compute *all* the GMVDs that are logically implied by an acyclic hypergraph structure, as the following theorem suggests.

Theorem 2 Let ϕ be a distribution defined on an hypergraph $\mathcal{H}$. Suppose X and Y are disjoint sets of attributes. Then the GMVD $X \multimap Y$ follows logically from ϕ if and only if Y is the union of some connected components of the hypergraph $\mathcal{H}$ with the set of nodes X deleted.

The proof of Theorem 2 can be derived in a similar fashion to the one in [8], and will be shown in a more complete paper.

Theorem 2 implies that the set of *all* GMVDs that can be logically inferred from an acyclic hypergraph structure $\mathcal{H}$ is given by:

$$\{X \multimap Y_1 \mid Y_2 \mid \ldots \mid Y_m \quad | \quad X \in \mathcal{L}\},$$

where $\mathcal{L}$ is the set of J-keys of $\mathcal{H}$ and the Y_i's are the disconnected components of $\mathcal{H}$ obtained by deleting the set of variables X. By Theorem 1, the set $\mathbf{X}$ of keys in the conflict-free cover of an input set of GMVDs is in fact equal to the set of J-keys of the constructed acyclic hypergraph structure $\mathcal{H}$. We can immediately conclude that $\mathcal{H}$ is a *perfect map*.

4 Conclusion

In this paper, a method for constructing the dependency structure of a Markov distribution is proposed. Our approach takes full advantage of a recent result showing nonembedded conditional independencies have a *complete* axiomatization. This result provides us with a tool for constructing a dependency structure which is a *perfect map* of the input set of nonembedded conditional independencies. That is, every nonembedded conditional independence logically implied by the input set can be inferred from the dependency structure, and every nonembedded conditional independence inferred from the dependency structure is logically implied by the input set. This technique involves determining a conflict-free cover of a given input set of nonembedded conditional independencies.

The problem of constructing the dependency structure of a Bayesian distribution given an input set of *both* embedded and nonembedded conditional independencies is currently being investigated.

References

[1] C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5(3):241–259, September 1980.

[2] C. Beeri, R. Fagin, and J.H. Howard. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 47–61, 1977.

[3] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, July 1983.

[4] C. Berge. *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1976.

[5] R. Bouckaert. Properties of bayesian belief network learning algorithms. In *Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109. Morgan Kaufmann Publishers, 1994.

[6] M. Druzdzel and L. van der Gaag. Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In *Eleventh*

Conference on Uncertainty in Artificial Intelligence, pages 141–148. Morgan Kaufmann Publishers, 1995.

[7] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems*, 2(3):262–278, September 1977.

[8] R. Fagin, A. Mendelzon, and J. Ullman. A simplified universal relation assumption and its properties. *ACM Transactions on Database Systems*, 7(3):343–360, September 1982.

[9] P. Hajek, T. Havranek, and R. Jirousek. *Uncertain Information Processing in Expert Systems*. CRC Press, 1992.

[10] J. Hu and Y. Xiang. Learning belief networks in domains with recursively embedded pseudo independent submodels. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 258–265. Morgan Kaufmann Publishers, 1997.

[11] F.V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, 1996.

[12] T.T. Lee. An information-theoretic analysis of relational databases-part ii: Information structures of database schemas. *IEEE Transactions on Software Engineering*, SE-13(10):1062–1072, 1987.

[13] Y. Edmund Lien. On the equivalence of database models. *Journal of the ACM*, 29(2):336–362, October 1982.

[14] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. Wiley, New York, 1990.

[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, California, 1988.

[16] Y. Sagiv. An algorithm for inferring multivalued dependencies with an application to propositional logic. *Journal of the ACM*, 27(2):250–262, April 1980.

[17] G. Shafer. An axiomatic study of computation in hypertrees. School of Business Working Papers 232, University of Kansas, 1991.

[18] M. Studeny. Conditional independence relations have no finite complete characterization. In *Eleventh Prague Conference on Information Theory, Statistical Decision Foundation and Random Processes*, 1990.

[19] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227. GE Corporate Research and Development, 1990.

[20] S.K.M. Wong. The relational structure of belief networks. *Submitted for publication*, 1997.

[21] S.K.M. Wong, C.J. Butz, and Y. Xiang. Automated database scheme design using mined data dependencies. *To appear in Journal of the American Society for Information Science*, 49(5), 1998.

[22] S.K.M. Wong and Z.W. Wang. On axiomatization of probabilistic conditional independence. In *Tenth Conference on Uncertainty in Artificial Intelligence*, pages 591–597. Morgan Kaufmann Publishers, 1994.

[23] S.K.M. Wong and Y. Xiang. Construction of a markov network from data for probabilistic inference. In *Third International Workshop on Rough Sets and Soft Computing*, pages 562–569, 1994.