Automated Database Schema Design Using Mined Data Dependencies

S.K.M. Wong, C.J. Butz and Y. Xiang
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
e-mail: {wong,butz,yxiang}@cs.uregina.ca
fax: (306)585-4745

Abstract

Data dependencies are used in database schema design to enforce the correctness of a database as well as to reduce redundant data. These dependencies are usually determined from the semantics of the attributes and are then enforced upon the relations. This paper describes a bottom-up procedure for discovering multivalued dependencies (MVDs) in observed data without knowing à priori the relationships amongst the attributes. The proposed algorithm is an application of the technique we designed for learning conditional independencies in probabilistic reasoning. A prototype system for automated database schema design has been implemented. Experiments were carried out to demonstrate both the effectiveness and efficiency of our method.

1 Introduction

Traditionally, data dependencies in relational databases (Maier, 1983) are constraints inferred by the database designer from the semantics of the attributes involved. These constraints are the rules that the data must obey in all instances to safeguard the correctness of the database. To reduce redundant data, these dependencies are also used for normalization by decomposing the database into an appropriate set of smaller relations. It should perhaps be emphasized that such a schema design is a top-down process based on the semantic dependencies amongst the attributes. Dependencies such as functional dependencies (FDs) and multivalued dependencies (MVDs) have been studied extensively (Beeri, Fagin & Howard, 1977; Delobel, 1978; Fagin, 1977) as they play a key role in the design of desirable database schemas.

In many applications, however, the exact relationships (i.e., data dependencies), determined by the semantics of the attributes in the database, may not be known à priori to the database designer. One may instead encounter a large collection of data for which no

information is known regarding the semantics of the attributes. Under these circumstances, the database designer would be unable to normalize such a database. It is therefore useful to develop an algorithm that is capable of mining dependencies in observed data. In this paper, we suggest a learning method for mining MVDs in observed data. Our system requires no à priori knowledge regarding the semantics of the attributes involved. The required input is simply a repository of observed data (preferably in a tabular form). Our system is capable of learning MVDs from the data and outputs a database schema encoding all the discovered MVDs. In fact, the output schema satisfies an acyclic join dependency (Wong, Xiang & Nie, 1994c; Wong, Butz & Xiang, 1995) which is known to possess several desirable properties (Beeri et al., 1983) in database applications. Thereby, not only do we mine dependencies which hold in the observed data, but we also determine a desirable database schema. Our method can be seen as a bottom-up approach for automated schema design.

Other researchers have also proposed other bottom-up techniques for discovering relational dependencies in data. Flach (1990) and Savnik (1993) suggested methods for mining FDs in observed data. A relation, however, decomposes losslessly if and only if the required MVD holds (Maier, 1983). FD is a sufficient but not a necessary condition for lossless decomposition. That is, MVDs are less restrictive than FDs and therefore more useful than FDs for database normalization. It should be noted that our learning algorithm discovers those MVDs that are logically implied by the corresponding FDs.

The proposed algorithm is a special application of the multi-link method (Xiang, Wong & Cercone, 1995) we developed for discovering probabilistic conditional independencies. This method is particularly suitable for our purposes here because we do not need to learn *embedded* conditional independencies as in other approaches such as the one proposed by Cooper and Herskovits (1992). We have adapted our algorithm for determining probabilistic conditional independencies in observed data to mining MVDs. This is possible because there exists an intriguing relationship between probabilistic reasoning systems and traditional relational database systems. In fact, MVD is equivalent to probabilistic conditional independence in a uniform distribution. While many researchers (Dechter, 1990; Hill, 1993; Lauritzen & Spiegelhalter, 1988; Lee, 1983; Pearl and Verma, 1987; Pearl, 1988) have noticed similarities between these two distinct but closely related knowledge systems, the relationship delves far beyond mere similarities. It has been shown that a Bayesian network can be represented as an extended relational data model (Wong, Xiang & Nie, 1994c). A probabilistic model (Hajek, Havranek & Jirousek, 1992; Neapolitan, 1990; Pearl, 1988) can actually be implemented as a generalized relational database (Wong, Butz & Xiang, 1995). Furthermore, the *Chase*, a relational technique for determining the implication of a set of data dependencies, can be modified to determine implication of probabilistic conditional independencies (Wong, 1996). In this paper, we demonstrate that a technique developed for probabilistic reasoning can be applied to relational databases.

This paper is organized as follows. In Section 2, we show why mining MVDs is more useful than mining FDs for database normalization. In Section 3, we demonstrate that a relation in a standard relational database can be viewed as uniform joint probability distribution. We use an example to illustrate informally that the notion of MVD is equivalent to that of probabilistic conditional independence when dealing with a uniform distribution. In Section 4, we formally show that in general (i.e. for any arbitrary distribution) MVD is a necessary

but not sufficient condition for probabilistic conditional independence. In Section 5, we describe how MVDs can be discovered from a relation by adopting the learning procedure we developed for probabilistic reasoning. In Section 6, we state our learning algorithm and illustrate its execution with an example. The run-time analysis of our algorithm and experimental results are presented in Section 7. Section 8 contains the conclusion.

2 FDs are Less Useful than MVDs in Database Normalization

In this section, we demonstrate why we search for MVDs instead of FDs. We will show that FDs are too restrictive a condition for decomposing some relations. For completeness, we begin by defining some basic notions in relational database theory.

Let $\mathcal{N}$ be a finite set of attributes. We will use the letter a with a subscript i, (i.e., a_i), to denote a single attribute and $\ldots, X, Y, Z$ to represent a subset of attributes of $\mathcal{N}$. Each attribute $a_i \in \mathcal{N}$ takes on values from a finite domain V_{a_i}. Consider a subset of attributes $X = \{a_1, a_2, ..., a_m\} \subseteq \mathcal{N}$. Let $V_X = V_{a_1} \cup V_{a_2} \cup ... \cup V_{a_m}$ be the domain of X. A X-tuple t_X is a mapping from X to V_X, i.e., $t_X : X \longrightarrow V_X$, with the restriction that for each attribute $a_i \in X$, $t_X[a_i]$ must be in V_{a_i}. (We write t instead of t_X if X is understood.) Thus t is a mapping that associates a value with each attribute in X. If Y is a subset of X and t is a X-tuple, then $t[Y]$ denotes the Y-tuple obtained by restricting the mapping to Y. Let $y = t[Y]$. We call y a Y-value which is also referred to as a *configuration* of Y. A X-relation r (or a relation r_X over X, or simply a relation r if X is understood), is a finite set of X-tuples or X-values. If r is a X-relation and Y is a subset of X, then by $r[Y]$, the projection of relation r onto Y, we mean the set of tuples $t[Y]$, where t is in r. (Note we will choose the notation, r, $r[R]$, or r_R, which is most convenient for a particular discussion.)

Definition 1 Let $r[R]$ be a relation with $X, Y \subseteq R$. We say that the *functional dependency* (FD) $X \rightarrow Y$ holds on relation $r[R]$ if for any two tuples $t_1, t_2 \in r[R]$ with $t_1[X] = t_2[X]$, then it is also the case that $t_1[Y] = t_2[Y]$.

The following example demonstrates how FDs can be used to decompose a relation.

Example 1 Consider the relation $r[R]$, where $R = \{a_1, a_2, a_3\}$, as shown in Figure 1. Note that the FD $\{a_1\} \rightarrow \{a_3\}$ holds on $r[a_1a_2a_3]$. This relation can be decomposed losslessly into two relations $r[a_1a_2]$ and $r[a_1a_3]$ as shown in Figure 2. That is, $r[a_1a_2a_3] = r[a_1a_2] \bowtie r[a_1a_3]$, where $\bowtie$ is the natural join operator in the standard relational database model. $\square$

The next example demonstrates that FDs are too restrictive a condition for decomposition.

Example 2 Consider the relation $r'[R]$ on schema $R = \{a_1, a_2, a_3\}$ in Figure 3. Note that neither of the FDs $\{a_2\} \rightarrow \{a_1\}$ or $\{a_2\} \rightarrow \{a_3\}$ hold in $r'[R]$. This relation, however, can still be decomposed losslessly onto the individual schemas $\{a_1, a_2\}$ and $\{a_2, a_3\}$ as depicted in Figure 4. That is, $r'[R] = r'[a_1a_2] \bowtie r'[a_2a_3]$. $\square$

$r[a_1a_2a_3] =$

a_1	a_2	a_3
0	0	0
0	1	0
1	0	1

Figure 1: A relation $r[a_1a_2a_3]$ satisfying the FD $\{a_1\} \rightarrow \{a_3\}$.

$r[a_1a_2] =$

a_1	a_2
0	0
0	1
1	0

,

$r[a_1a_3] =$

a_1	a_3
0	0
1	1

Figure 2: The projections $r[a_1a_2]$ and $r[a_1a_3]$ of $r[a_1a_2a_3]$.

Example 2 clearly shows that the use of FDs for decomposition may not reveal all the valid decompositions. It has long been proven that the necessary and sufficient condition for lossless decomposition of relations is MVD (Maier, 1983). Thereby, it is more useful to discover MVDs, rather than FDs, in observed data.

3 Relations versus Probability Distributions

Our objective here is to show informally that a relation can be viewed as a uniform joint probability distribution. In this context, we will use a simple example to illustrate the close relationship between MVD and probabilistic conditional independence. A formal examination of the relationship between MVD and probabilistic conditional independence can be found in (Wong, 1994a).

Definition 2 Let $r[R]$ be a relation with $X, Y \subseteq R$. We say that the *multivalued dependency* (MVD) $X \twoheadrightarrow Y$ holds on relation $r[R]$ if for any $t_1, t_2 \in r$ with $t_1[X] = t_2[X]$, there exists a tuple $t_3 \in r$ such that $t_3[XY] = t_1[XY]$ and $t_3[R - XY] = t_2[R - XY]$. By XY, we mean $X \cup Y$.

$r'[a_1a_2a_3] =$

a_1	a_2	a_3
0	0	0
1	0	0
0	1	0
0	1	1

Figure 3: A relation $r'[a_1a_2a_3]$ not satisfying either of the FDs $\{a_2\} \rightarrow \{a_1\}$ or $\{a_2\} \rightarrow \{a_3\}$.

$r'[a_1a_2] =$

a_1	a_2
0	0
1	0
0	1

,

$r'[a_2a_3] =$

a_2	a_3
0	0
1	0
1	1

Figure 4: The projections $r'[a_1a_2]$ and $r'[a_2a_3]$ of $r'[a_1a_2a_3]$.

The implication of Definition 2 is that if a MVD $X \twoheadrightarrow Y$ holds in a relation $r[R]$, then $r[R]$ can be decomposed *losslessly* into $r[XY]$ and $r[XZ]$ where Z is $R - XY$. That is, $r[R] = r[XY] \bowtie r[XZ]$. It follows from the definition of MVD that if $X \twoheadrightarrow Y$, then $X \twoheadrightarrow Z$, where Z is $R - XY$. Thus, $X \twoheadrightarrow Y$ may also be written as $X \twoheadrightarrow Y \mid Z$. We illustrate this concept with the aid of an example.

Example 3 Consider the observed data represented by a relation $r[a_1a_2a_3a_4a_5]$ as shown in Figure 5. It can be easily verified that the relation $r[a_1a_2a_3a_4a_5]$ satisfies the following MVDs:

$$a_2 \twoheadrightarrow a_1 \mid a_3\ a_4\ a_5, \tag{1}$$

$$a_3 \twoheadrightarrow a_1\ a_2 \mid a_4\ a_5. \tag{2}$$

Using these MVDs, we can decompose the relation $r[a_1a_2a_3a_4a_5]$ *losslessly* into three smaller relations $r[a_1a_2]$, $r[a_2a_3]$ and $r[a_3a_4a_5]$, as shown in Figure 6, i.e., $r[a_1a_2a_3a_4a_5] = r[a_1a_2] \bowtie r[a_2a_3] \bowtie r[a_3a_4a_5]$. $\square$

(Note that we sometimes write $\{a_i\} \twoheadrightarrow \{a_j\}$ as $a_i \twoheadrightarrow a_j$.)

$r[a_1a_2a_3a_4a_5] =$

a_1	a_2	a_3	a_4	a_5
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	1	0	1
1	0	1	1	0

Figure 5: A relation $r[a_1a_2a_3a_4a_5]$ representing the observed data.

On the other hand, one can view the relation $r[a_1a_2a_3a_4a_5]$ in Figure 5 as a *uniform joint probability distribution* $p_R(a_1, a_2, a_3, a_4, a_5)$ depicted in Figure 7. The subscript R in p_R indicates that the distribution is over R. Before introducing the concept of probabilistic conditional independence, we first define the notions of marginalization and conditional probability.

$r[a_1a_2a_3a_4a_5] =$

a_1	a_2
0	1
1	0

$\bowtie$

a_2	a_3
1	0
1	1
0	1

$\bowtie$

a_3	a_4	a_5
0	0	0
0	1	0
1	0	1
1	1	0

Figure 6: Lossless decomposition of the relation in Figure 5, namely $r[a_1a_2a_3a_4a_5] = r[a_1a_2] \bowtie r[a_2a_3] \bowtie r[a_3a_4a_5]$.

$p_R(a_1, a_2, a_3, a_4, a_5) =$

a_1	a_2	a_3	a_4	a_5	$f_{p_R(a_1,a_2,a_3,a_4,a_5)}$
0	1	0	0	0	1/6
0	1	0	1	0	1/6
0	1	1	0	1	1/6
0	1	1	1	0	1/6
1	0	1	0	1	1/6
1	0	1	1	0	1/6

Figure 7: The uniform joint probability distribution $p_R(a_1, a_2, a_3, a_4, a_5)$ corresponding to the relation $r[a_1a_2a_3a_4a_5]$ in Figure 5.

Let V_{a_i} denote the domain of the variable $a_i \in \mathcal{N}$. (Note that the probabilistic reasoning term *variable* will be referred to as *attribute*.) If X and Y are disjoint subsets of $\mathcal{N}$ with $\mathbf{c}_X$ denoting a configuration (tuple) of X and $\mathbf{c}_Y$ denoting a configuration of Y, i.e., $\mathbf{c}_X \in V_X$ and $\mathbf{c}_Y \in V_Y$, we then write $\mathbf{c}_X * \mathbf{c}_Y$ to denote the configuration of $X \cup Y$ obtained by *concatenating* $\mathbf{c}_X$ and $\mathbf{c}_Y$. That is, $\mathbf{c}_X * \mathbf{c}_Y$ is the unique configuration of $X \cup Y$ such that $(\mathbf{c}_X * \mathbf{c}_Y)[X] = \mathbf{c}_X$ and $(\mathbf{c}_X * \mathbf{c}_Y)[Y] = \mathbf{c}_Y$.

Definition 3 Consider a real valued function p_X on a set X of attributes. If $Y \subseteq X$, then the *marginal* distribution of p_X on Y, denoted p_Y, is the function on Y defined by:

$$p_Y(\mathbf{c}_Y) = \sum_{\mathbf{c}_{X-Y}} p_X(\mathbf{c}_Y * \mathbf{c}_{X-Y}),$$

where $\mathbf{c}_Y$ is a configuration of Y, and $\mathbf{c}_{X-Y}$ is a configuration of $X - Y$ after removing from X the values of the attributes that are also in Y. Clearly, $\mathbf{c}_Y * \mathbf{c}_{X-Y}$ is a configuration of X.

For example, the marginal distribution $p_Y(a_1, a_2)$ on $Y = \{a_1, a_2\}$ of the uniform distribution in Figure 7 is depicted in Figure 8.

To simplify notation, henceforth, we may write p_Y as p whenever Y is understood. We may, however, write p_Y to emphasize that the distribution is over Y. Let $X, Y \subseteq \mathcal{N}$ be two disjoint sets and p be a distribution over $\mathcal{N}$. The conditional probability of X conditioned on Y is defined as follows:

$$p(X \mid Y) = \frac{p(XY)}{p(Y)}.$$

$p_Y(a_1, a_2) =$

a_1	a_2	$f_{p_Y(a_1,a_2)}$
0	1	4/6
1	0	2/6

Figure 8: The marginal distribution $p_Y(a_1, a_2)$ of the uniform joint probability distribution $p_R(a_1, a_2, a_3, a_4, a_5)$ in Figure 7.

Definition 4 Let $X, Y, Z \subseteq \mathcal{N}$ and p be a distribution over $\mathcal{N}$. Y is *probabilistically conditionally independent* of Z given X if

$$p(Y \mid XZ) = p(Y \mid X).$$

The same conditional independence can also be expressed by

$$p(XYZ) = \frac{p(XY) \cdot p(XZ)}{p(X)},$$

or

$$p(YZ \mid X) = p(Y \mid X) \cdot p(Z \mid X).$$

It is not difficult to see that the uniform joint distribution $p(a_1, a_2, a_3, a_4, a_5)$, depicted in Figure 7, satisfies the probabilistic conditional independencies, namely:

$$p(a_1, a_2, a_3, a_4, a_5) = \frac{p(a_1, a_2) \cdot p(a_2, a_3, a_4, a_5)}{p(a_2)}, \tag{3}$$

and

$$p(a_1, a_2, a_3, a_4, a_5) = \frac{p(a_1, a_2, a_3) \cdot p(a_3, a_4, a_5)}{p(a_3)}, \tag{4}$$

where $p(a_2), p(a_3), p(a_1, a_2), p(a_1, a_2, a_3), p(a_3, a_4, a_5)$ and $p(a_2, a_3, a_4, a_5)$ are the marginal distributions of $p(a_1, a_2, a_3, a_4, a_5)$. For instance, using equation (3), we verify the configuration $(0, 1, 0, 0, 0)$ as follows:

$$p(0, 1, 0, 0, 0) = \frac{p(0, 1) \cdot p(1, 0, 0, 0)}{p(1)} = \frac{(4/6) \cdot (1/6)}{4/6} = \frac{1}{6}.$$

One can easily verify that the relation $r[a_1a_2a_3a_4a_5]$ satisfies the MVDs defined by equations (1) and (2) if and only if the uniform distribution $p(a_1, a_2, a_3, a_4, a_5)$ satisfies the corresponding probabilistic conditional independencies defined by equations (3) and (4).

Thus, the above example clearly demonstrates that the notion of multivalued dependency in relational databases is closely connected to that of probabilistic conditional independence in Bayesian networks (Jensen, 1996; Pearl, 1988). We can take advantage of this relationship by applying our algorithm for learning probabilistic conditional independencies to one which learns multivalued dependencies. This is achieved by supplying observed data, representing a uniform distribution, as input. This will be described in detail in Section 5.

4 Multivalued Dependency versus Probabilistic Independence

The following demonstrates that MVD is a subclass of probabilistic conditional independence. In Section 5, we will show how such a relationship facilitates the discovery of MVDs in a knowledge system.

MVDs can also be understood in terms of sorting and counting. Let r_R denote a relation over schema $R = \{a_1, a_2, \ldots, a_s\}$. We can define a function $n_W[X = x]$ to count the number of distinct W-values associated with a given X-value in a relation r_R:

$$n_W[X = x](r_R) = \mid \{t[W] \mid t \in r_R,\ t[X] = x\} \mid, \tag{5}$$

where $\mid \cdot \mid$ denotes the cardinality of a set and X, W are arbitrary subsets of R. It is important that the reader understand equation (5) or confusion may arise during the rest of this section. Equation (5) can alternatively be expressed as:

$$n_W[X = x](r_R) = \mid \ \pi_W(\sigma_{X=x}(r_R)) \ \mid,$$

where π, σ are the *projection* and *selection* operators in the standard relational data model.

Example 4 For the relation $r[a_1a_2a_3a_4a_5]$ in Figure 5,

$$\begin{aligned}
&n_{a_1a_2a_3a_4a_5}[a_1a_2 = 01](r[a_1a_2a_3a_4a_5]) = 4,\\
&n_{a_4}[a_1a_2 = 01](r[a_1a_2a_3a_4a_5]) = 2,\\
&n_{a_3}[a_1a_2 = 10](r[a_1a_2a_3a_4a_5]) = 1. \quad \square
\end{aligned}$$

It can be verified that relation a r_R satisfies the MVD, $X \twoheadrightarrow Y$, if and only if for any X-value x in r_R,

$$n_R[X = x](r_R) = n_{YX}[X = x](r_R) \cdot n_{XZ}[X = x](r_R), \tag{6}$$

where $Z = R - YX$. By definition, $n_{XW}[X = x] = n_W[X = x]$. Thus, equation (6) can be written as:

$$n_R[X = x](r_R) = n_Y[X = x](r_R) \cdot n_Z[X = x](r_R).$$

This implies that $X \twoheadrightarrow Y$ holds, if and only if, for every X-value x and Y-value y in r such that $yx = t[YX]$ and $t \in r$,

$$n_Z[X = x](r_R) = n_Z[YX = yx](r_R).$$

Likewise, we obtain:

$$n_Y[X = x](r_R) = n_Y[XZ = xz](r_R).$$

The above results about MVDs are summarized in the following lemma.

Lemma 1 (Maier, 1983) Let r_R be a relation over R, and let X and Y be disjoint subsets of R and $Z = R - YX$. Relation r_R satisfies the MVD $X \twoheadrightarrow Y$, if and only if for every X-value x, Y-value y, and Z-value z in r_R such that the YXZ-value yxz appears in r_R:

$$n_R[X = x](r_R) = n_Y[XZ = xz] \cdot n_Z[YX = yx]. \tag{7}$$

At this point, let us focus on the concept of joint probability distributions. Let $p_R(a_1, a_2, \ldots, a_s)$, where $R = \{a_1, a_2, \ldots, a_s\}$, denote a joint probability distribution. In particular, we are interested in a *uniform* distribution which we can view as a relation $r_{R \cup \{f_{p_R}\}}$ over the set of attributes $R \cup \{f_{p_R}\}$. An example of such a distribution is given in Figure 3. It should be noted that the terms of relation and distribution will be used interchangeably, if no confusion arises.

$p \;=\; r_{R \cup \{f_p\}} =$

a_1	a_2	a_3	f_p
0	0	0	α
0	1	0	α
0	1	1	α
1	1	0	α
1	1	1	α

Figure 9: An example of a uniform joint probability distribution p depicted as a relation $r_{R \cup \{f_p\}}$ over the set of attributes $R \cup \{f_p\} = \{a_1, a_2, a_3, f_p\}$.

Theorem 1 (Wong, 1994a) Let $p(a_1, a_2, \ldots, a_s)$ be a *uniform* joint probability distribution and let X and Y be disjoint subsets of R and $Z = R - YX$. The distribution p satisfies the probabilistic conditional independence condition:

$$p(xyz) = \frac{p(yx) \cdot p(xz)}{p(x)}, \tag{8}$$

where $x = t[X], yx = t[YX], xz = t[XZ],$ and $yxz = t[YXZ]$, if and only if the relation $r_R = r_{R \cup \{f_p\}}[R]$ satisfies the multivalued dependency $X \twoheadrightarrow Y$.

Proof: By the definition of marginalization,

$$p(yx) = \sum_{t_Z} p(t_{YX} * t_Z) = \sum_z p(yxz),$$

where $z = t_Z = t[Z]$ and $yxz = t[YXZ] = t[YX] * t[Z] = t_{YZ} * t_Z$.

Since p is a uniform function, i.e., $p(t) = \alpha$ for any configuration $t \in r_R$, by definition it immediately follows:

$$p(yx) = \sum_z p(yxz) = \alpha \cdot n_Z[YX = yx](r_R), \tag{9}$$

where $n_Z[YX = yx](r_R)$ is the number of distinct Z-values for a given YX-value yx in r_R. Similarly,

$$\begin{aligned} p(xz) &= \sum_{t_Y} p(t_Y * t_{XZ}) = \sum_{y} p(yxz) \\ &= \alpha \cdot n_Y[XZ = xz](r_R), \end{aligned} \tag{10}$$

where $n_Y[XZ = xz](r_R)$ is the number of distinct Y-values for a given XZ-value xz in r_R. Also,

$$\begin{aligned} p(x) &= \sum_{t_{YZ}} p(t_Y t_X t_Z) = \sum_{yz} p(yxz) \\ &= \alpha \cdot n_R[X = x](r_R), \end{aligned} \tag{11}$$

where $n_R[X = x](r_R)$ is the number of distinct tuples for a given X-value x in r_R.

Thus, from equations (8), (9), (10), and (11), we immediately obtain the desired result:

$$\frac{p(yx) \cdot p(xz)}{p(x)} = \alpha \cdot (\frac{n_Z[YX = yx](r_R) \cdot n_Y[XZ = xz](r_R)}{n_R[X = x](r_R)}). \qquad \square \tag{12}$$

(Note that in the above theorem, we have omitted the subscript denoting the domain of the distribution, i.e., p_X as p.) Consider, for example, the uniform probability distribution in Figure 9. It can be easily verified in this example that the MVDs $a_2 \twoheadrightarrow a_1$ and $a_2 \twoheadrightarrow a_3$ are satisfied by the relation $p[R]$ as shown in Figure 10. An equivalent condition that must hold is:

$$p(a_1, a_2, a_3) = \frac{p(a_1, a_2) \cdot p(a_2, a_3)}{p(a_2)},$$

where the marginal distributions, $p(a_2), p(a_1, a_2)$, and $p(a_2, a_3)$ of this distribution $p(a_1, a_2, a_3)$ are shown in Figure 11. The required result is obtained as shown in Figure 12.

$p[R] =$

a_1	a_2	a_3
0	0	0
0	1	0
0	1	1
1	1	0
1	1	1

$=$

a_1	a_2
0	0
0	1
1	1

$\bowtie$

a_2	a_3
0	0
1	0
1	1

Figure 10: The relation $p[R]$.

To conclude this section, we would like to reiterate the fact that in a uniform distribution, MVD holds if and only if the corresponding probabilistic conditional independence holds. In an arbitrary distribution, however, MVD is a necessary but *not* a sufficient condition for the existence of probabilistic conditional independence. That is, probabilistic conditional independence indeed implies MVD, but the converse is false.

$p(a_2) =$

a_2	$f_{p_{\{a_2\}}}$
0	α
1	4α

, $p(a_1, a_2) =$

a_1	a_2	$f_{p_{\{a_1,a_2\}}}$
0	0	α
0	1	2α
1	1	2α

, $p(a_2, a_3) =$

a_2	a_3	$f_{p_{\{a_2,a_3\}}}$
0	0	α
1	0	2α
1	1	2α

.

Figure 11: Marginal distributions $p(a_2), p(a_1, a_2)$, and $p(a_2, a_3)$ of the distribution $p(a_1, a_2, a_3)$ in Figure 9.

$$p(a_1, a_2, a_3) = \frac{p(a_1, a_2) \cdot p(a_2, a_3)}{p(a_2)} =$$

a_1	a_2	a_3	$f_{p_{\{a_1,a_2,a_3\}}}$
0	0	0	α
0	1	0	α
0	1	1	α
1	1	0	α
1	1	1	α

.

Figure 12: Resultant relation of the computation $p(a_1, a_2) \cdot p(a_2, a_3)/(p(a_2)$.

5 Discovery of Multivalued Dependencies

We have shown in the previous section that, in a uniform distribution, MVD is equivalent to probabilistic conditional independence. Thus, we can directly apply the method originally proposed for learning probabilistic conditional independencies from a joint distribution to discover multivalued dependencies in observed raw data. Before discussing our learning algorithm, let us first introduce the notions of *hypertree* (Shafer, 1991) and *Markov* distributions (Hajek, Havranek & Jirousek, 1992).

Let $\mathcal{N} = \{a_1, a_2, ..., a_m\}$ denote a set of attributes. We say that $\mathcal{G}$ is a *hypergraph*, if $\mathcal{G}$ is a subset of the power set $2^{\mathcal{N}}$. An element h in a hypergraph $\mathcal{G}$ is a *twig* if there exists another element g in $\mathcal{G}$, distinct from h, such that $h \cap (\cup(\mathcal{G} - \{h\})) = h \cap g$. We call any such g a *branch* for the twig h. A hypergraph is a *hypertree* if its elements (hyperedges) can be ordered, say $h_1, h_2, ..., h_i$, so that h_i is a twig in $\{h_1, h_2, ..., h_i\}$, for $i = 2, ..., n$. We call any such ordering a *hypertree construction ordering* for $\mathcal{G}$. Given a hypertree construction ordering $h_1, h_2, ..., h_n$, we can choose, for i from 2 to n, an integer $b(i)$ such that $1 \leq b(i) \leq i - 1$ and $h_{b(i)}$ is a branch for h_i in $\{h_1, h_2, ..., h_i\}$. We call the function $b(i)$ satisfying this condition a *branching function* for $\mathcal{G}$ and the hypertree construction ordering $h_1, h_2, ..., h_n$. A hypertree is also referred to as an *acyclic* hypergraph in relational database theory. An acyclic hypergraph is characterized by the chordal and conformal properties. A graph is *chordal* if every cycle with at least four distinct nodes has a chord, that is, and edge connecting two nonconsecutive nodes on the cycle. A hypergraph $\mathcal{G}$ is *conformal* if for every maximal clique V in the undirected graph, there is a hyperedge of $\mathcal{G}$ which contains V.

Consider, for example, the following joint probability distribution:

$$p(a_1, a_2, a_3, a_4, a_5, a_6) = \frac{p(a_1, a_2, a_3) \cdot p(a_1, a_2, a_4) \cdot p(a_2, a_3, a_5) \cdot p(a_5, a_6)}{p(a_1, a_2) \cdot p(a_2, a_3) \cdot p(a_5)}. \tag{13}$$

The probabilistic conditional independencies of this distribution can be conveniently depicted by an undirected graph as shown in Figure 13.

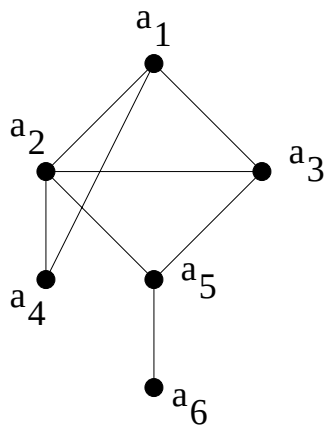

Figure 13: The undirected graph depicting the conditional independencies of the probability distribution defined by equation (13).

In many situations, it is more convenient to characterize probabilistic conditional independencies by a hypergraph. For example, the hypergraph $\mathcal{G} = \{h_1, h_2, h_3, h_4\}$ corresponding to the undirected graph in Figure 13 is shown in Figure 14, where $h_1 = \{a_1, a_2, a_3\}$, $h_2 = \{a_1, a_2, a_4\}$, $h_3 = \{a_2, a_3, a_5\}$, and $h_4 = \{a_5, a_6\}$. Each hyperedge in $\mathcal{G}$ is in fact a *maximal clique* of the undirected graph in Figure 13. We say that the distribution $p(a_1, a_2, a_3, a_4, a_5, a_6)$ defined by equation (13) is *factorized* on this hypergraph.

Lemma 2 If a joint probability distribution p_R on a finite set R of attributes is factorized on an acyclic hypergraph (a hypertree) $\mathcal{G} = \{R_1, R_2, \ldots, R_n\}$, then

$$p_R = \frac{\prod_{i=1}^{n} p_{R_i}}{\prod_{i=2}^{n} p_{R_{b(i)} \cap R_i}} = \frac{\prod_{i=1}^{n} p_{R_i}}{\prod_{i=2}^{n} p_{R_{i^*} \cap R_i}}, \tag{14}$$

where $i^* = b(i)$, and p_{R_i}, $p_{R_{i^*} \cap R_i}$ are marginal distributions of p_R.

We call a distribution defined by equation (14) a Markov distribution (Hajek, Havranek & Jirousek, 1992; Neapolitan, 1990).

By representing a joint probability distribution as a Markov distribution, it can been seen from Figure 13 that certain embedded independencies such as

$$p(a_1, a_2, a_3) \quad = \quad \frac{p(a_1, a_2) \cdot p(a_1, a_3)}{p(a_1)}$$

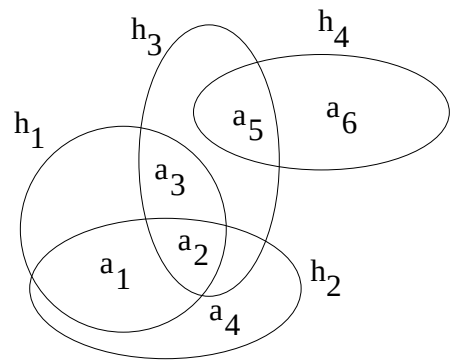

Figure 14: The hypergraph $\mathcal{G} = \{h_1, h_2, h_3, h_4\}$ corresponding to the undirected graph in Figure 13.

cannot be expressed. In contrast, this embedded conditional independence is reflected in a Bayesian distribution (Pearl, 1988).

Note that probabilistic conditional independencies can be easily inferred from the undirected graph (the hypertree structure) depicting a Markov distribution. For example, from Figure 13, we obtain:

$$\begin{aligned} p(a_3, a_4, a_5, a_6 | a_1, a_2) &= p(a_4 | a_{1,2}) \cdot p(a_3, a_5, a_6 | a_1, a_2), \\ p(a_1, a_4, a_5, a_6 | a_2, a_3) &= p(a_1, a_4 | a_2, a_3) \cdot p(a_5, a_6 | a_2, a_3), \\ p(a_1, a_2, a_3, a_4, a_6 | a_5) &= p(a_1, a_2, a_3, a_4 | a_5) \cdot p(a_6 | a_5). \end{aligned}$$

In the next section, we will demonstrate that a Markov distribution can be characterized by a entropy function. Minimizing the entropy function will then be used to learn the dependency structure of the Markov distribution in Section 5.2.

5.1 Characterization of a Markov Distribution by an Entropy Function

Given a distribution p_X on $X \subseteq R$, we can define a function as follows:

$$\begin{aligned} \mathcal{H}(p_X) &= -\sum_{t_X} p_X(t_X) \log p_X(t_X) \\ &= -\sum_{x} p_X(x) \log p_X(x), \end{aligned} \tag{15}$$

where t is a tuple (configuration) of X and $x = t_X = t[X]$ is a X-value, and $\mathcal{H}$ is the Shannon entropy.

From a Markov distribution,

$$p_R = \frac{\prod_{i=1}^{n} p_{R_i}}{\prod_{i=2}^{n} p_{R_{i^*} \cap R_i}} = \frac{\prod_{i=1}^{n} p_{R_i}}{\prod_{i=2}^{n} p_{R'_i}}, \tag{16}$$

where R_{i^*} is the branch of R_i, and $R' = R_{i^*} \cap R_i$, $i = 2, 3, \ldots, n$, we obtain:

$$\begin{aligned}
\mathcal{H}(p_R) &= -\sum_{t_R} p_R(t_R) \log p_R(t_R) \\
&= -\sum_{t_R} p_R(t_R) \log p_{R_1}(t_{R_1}) - \cdots - \sum_{t_R} p_R(t_R) \log p_{R_n}(t_{R_n}) \\
&\quad + \sum_{t_R} p_R(t_R) \log p_{R'_2}(t_{R'_2}) + \cdots + \sum_{t_R} p_R(t_R) \log p_{R'_n}(t_{R'_n}) \\
&= -\sum_{i=1}^{n} p_{R_i} \log p_{R_i} + \sum_{i=2}^{n} p_{R'_i} \log p_{R'_i} \\
&= \sum_{i=1}^{n} \mathcal{H}(p_{R_i}) - \sum_{i=2}^{n} \mathcal{H}(p_{R_{i^*} \cap R_i}). \qquad (17)
\end{aligned}$$

As we only consider Markov distributions in the following exposition, we write $\mathcal{H}(p_X)$ as $\mathcal{H}(X)$ if no confusion arises. Thus, equation (17) can be written as:

$$\mathcal{H}(R) = \sum_{i=1}^{n} \mathcal{H}(R_i) - \sum_{i=2}^{n} \mathcal{H}(R_{i^*} \cap R_i), \qquad (18)$$

where $R = R_1 R_2 \ldots R_n$.

Let $R = XYZ$ and let X, Y, Z be disjoint subsets of R. For a Markov distribution with $n = 2$, we have:

$$p_{XYZ} = \frac{p_{XY} \cdot p_{XZ}}{p_X}, \qquad (19)$$

which means that Y and Z are conditionally independent given X. By equation (18),

$$\mathcal{H}(p_{XYZ}) = \mathcal{H}(p_{XY}) + \mathcal{H}(p_{XZ}) - \mathcal{H}(p_X)$$

or

$$\mathcal{H}(XYZ) = \mathcal{H}(XY) + \mathcal{H}(XZ) - \mathcal{H}(X). \qquad (20)$$

The above results are summarized in the following theorem.

Theorem 2 Let $\mathcal{G} = \{R_1, R_2, \ldots, R_n\}$ be hypertree and $R = R_1 R_2 \ldots R_n$. Let the sequence R_1, R_2, $\ldots R_n$ be a tree construction ordering for $\mathcal{G}$ such that $(R_1 R_2 \ldots R_{i-1}) \cap R_i = R_{i^*} \cap R_i$ for $1 \leq i^* \leq n-1$ and $2 \leq i \leq n$. A joint probability distribution p_R factorized on $\mathcal{G}$ is a Markov distribution, if and only if

$$\mathcal{H}(p_R) = \sum_{i=1}^{n} \mathcal{H}(p_{R_i}) - \sum_{i=2}^{n} \mathcal{H}(p_{R_{i^*} \cap R_i}). \qquad (21)$$

This theorem indicates that we can characterize a Markov distribution by an entropy function. In the next section, we will demonstrate how we learn the dependency structure of a Markov distribution.

5.2 Learning the Dependency Structure of a Markov Distribution

The proposed method for learning the dependency structure of a Markov distribution is based on minimizing the Kullback-Leibler $I(p, p')$ cross-entropy (Kullback & Leibler, 1951), a measure of closeness between a true distribution p and an approximate distribution p'. The cross entropy is defined by:

$$I(p, p') = \sum_x p(x) \log \frac{p(x)}{p'(x)},$$

where x is a configuration (tuple) of the set of attributes $R = \{a_1, a_2, \ldots, a_s\}$. With a fixed p, we can choose, from the set of all possible Markov distributions $\{p'\}$, the distribution p_0 that minimizes $I(p, p')$. For a Markov distribution, the cross-entropy can be written as:

$$I(p, p') = \sum_x p(x) \log p(x) \quad - \quad \sum_x p'(x) \log p'(x) \quad = \mathcal{H}(p') \; - \; \mathcal{H}(p),$$

where $\mathcal{H}$ is the Shannon entropy as defined by equation (15). Thus, for a given p, minimizing $I(p, p')$ is equivalent to minimizing the entropy $\mathcal{H}(p')$. That is,

$$\min_{p'' \in \{p'\}}(I(p, p'')) = \min_{p'' \in \{p'\}}(\mathcal{H}(p'')). \tag{22}$$

An approximate method (Wong & Xiang, 1994b) for computing p_0 is outlined as follows. Initially, we may assume that all the attributes are probabilistically independent, i.e., there exists no edge between any two nodes (attributes) in the undirected graph representing the Markov distribution. Then an edge is added to the graph subject to the restriction that the resultant hypergraph must be a hypertree. The undirected graph of the Markov distribution with minimum entropy is being selected as the graph for further addition of other edges. This process is repeated until a predetermined *threshold*, which defines the rate of decrease of entropy between successive distributions, is reached. The output of this iterative procedure is the dependency structure of the *minimum* Markov distribution p_0. That is, p_0 satisfies both equations (21) and (22). From the output hypertree, we can infer the probabilistic conditional independencies which are satisfied by p_0.

6 A Lookahead Learning Algorithm

Our proposed algorithm is based on a multi-link technique (Xiang, Wong & Cercone, 1996) used for learning probabilistic conditional independencies in probabilistic reasoning. There are many possible techniques (Cooper & Herskovits, 1992; Heckerman, Geiger & Chickering, 1995; Herskovits & Cooper, 1990; Lam & Bacchus, 1994; Rebane, 1987; Spirtes & Glymour, 1991) for learning a Bayesian distribution from observed data. As already mentioned, a Bayesian distribution involves embedded conditional independencies, whereas a Markov distribution does not. Our method is simpler as we are primarily interested in discovering MVDs and not embedded MVDs. As demonstrated in Section 4, MVD is a necessary condition for a corresponding probabilistic conditional independence. Thereby, learning conditional independencies implies learning MVDs. In this section, we will present our learning algorithm and provide a simple example.

Since our focus is now shifting from theoretical analysis of the minimum entropy search to its practical implication, some assumptions on the context in which the algorithm is applied are imposed. The attributes used in describing the observed data have discrete domains. Furthermore, there are no tuples in the observed data with missing values. These two assumptions are made in most algorithms for learning probabilistic networks.

Before we present our algorithm, we first discuss the variables used in the algorithm. The variable *entropy-decrement-pass* is the calculated entropy decrement for that particular pass of a level. The largest entropy decrement value for all the passes of one level is saved in the variable *entropy-decrement-level*. Similarly, *G-pass* is the constructed undirected graph for each pass, while we use *G-level* to save the graph corresponding to *entropy-decrement-level*.

Algorithm 1

```
Input: A database D over a set 𝒩 of attributes.
begin
        initialize an empty graph G = (𝒩, E);
        G-level := G;
        repeat % each level consists of passes
                initialize the entropy decrement entropy-decrement-level := 0;
                for each edge e, do % each pass consists of steps
                    if G-pass = (𝒩, E ∪ e) is chordal then
                      construct the corresponding Markov distribution;
                      compute the entropy decrement entropy-decrement-pass locally;
                    if entropy-decrement-pass > entropy-decrement-level, then
                      entropy-decrement-level := entropy-decrement-pass;
                      G-level := G-pass;
                      save the corresponding Markov distribution;
                if entropy-decrement-level > 0 then
                    G := G-level;
                    save the corresponding Markov distribution;
                    done := false;
                else
                    done := true;
        until done = true;
        return G;
end
```

(Note that by testing for chordality and enforcing the conformal property in each pass, we can ensure that the output of the algorithm is a Markov distribution.)

For computational efficiency, we may simplify algorithm 1 by using single links. This will considerably speed up the search for MVDs. The following discussion will focus on single-link techniques. (The algorithm analysis, in Section 7, will be derived on the more general multi-link technique.) It is perhaps worth while to mention that single-link methods may

not correctly learn certain types of probabilistic models (Xiang, Wong & Cercone, 1996).

Algorithm 1 is illustrated with the aid of an example. Learning starts with a completely disconnected Markov network. This means that all the attributes are initially assumed to be probabilistically independent of each other. At each step of the search, each possible single link is added to the current network, and the entropy of the resulting Markov network is computed. Note that for n unconnected nodes, there are $O(n^2)$ single links. The network yielding the lowest entropy is chosen to be the starting network entering the next step of the search. After each step of the search, the decrement of the entropy is checked against a predetermined threshold. If the decrement is less than the threshold, the learning process terminates. Intuitively, the threshold value can be thought of as the belief in how good the sample is. Thus, for determining MVDs using a given relation (with no duplicates) the threshold can be set to zero. On the other hand, if a large set of tuples are given as input (with duplicates) then the threshold can be set accordingly. More specifically, the threshold should be set according to the size of the sample set (i.e., the smaller the sample set, the larger the threshold). The intuition is that when the sample set is smaller, erroneous dependencies may be introduced and hence erroneous links may be added to the network. Thus, by using a larger threshold, these *false* dependencies can be suppressed. Before we present the experimental results in the next section, we use an example to outline the search process. We remind the reader that Algorithm 1 consists of a number of levels and each level consists of a number of passes.

Example 5 Suppose we have a database D consisting of the observed data of a set of four attributes, $\mathcal{N} = \{a_1, a_2, a_3, a_4\}$, containing five tuples as shown in Figure 15. We have set the threshold to zero, the maximum size of a clique $\eta = 4$, and the maximum number of lookahead links to one. The MVDs which are known to hold in the database serve as a control set. In this example, the control set is $\{a_1 \twoheadrightarrow a_2,\ a_1 \twoheadrightarrow a_3,\ a_1 \twoheadrightarrow a_4,\ a_4 \twoheadrightarrow a_3\}$.

a_1	a_2	a_3	a_4
0	0	0	0
1	1	1	0
1	1	0	0
0	0	1	0
2	1	1	1

Figure 15: Observed data consisting of 4 attributes and 5 tuples.

Algorithm 1 starts with an empty undirected graph G, that is $G = (\mathcal{N}, E = \{\})$. In order to specify the graph G throughout the example, we will adopt the denotation of G by G_e^i where i is the level of the algorithm and e is the test edge added.

Level 1, Pass 1: The first edge added will be (a_1, a_2). The resultant graph $G^1_{(a_1,a_2)}$ is shown in Figure 16. With respect to $G^1_{(a_1,a_2)}$, marginal distributions are determined as depicted in

Figure 17. The corresponding Markov distribution is then constructed by:

$$p'(a_1, a_2, a_3, a_4) = p(a_1, a_2) \cdot p(a_3) \cdot p(a_4),$$

as shown in Figure 18. Using equation (15), the corresponding entropy $\mathcal{H}(p(a_1, a_2) \cdot p(a_3) \cdot p(a_4))$ is calculated to be 0.9673925. (Note that the details of the numerical calculations are shown in Figure 28.) At this point, one pass of the first level has completed.

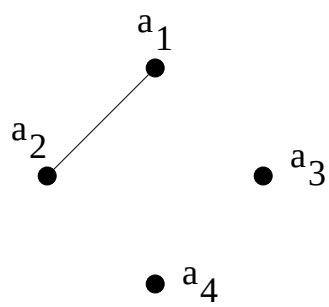

Figure 16: Level 1, Pass 1: The undirected graph $G^1_{(a_1,a_2)}$ constructed by adding the edge (a_1, a_2).

$p(a_1, a_2)$ =

a_1	a_2	$f_{p(a_1,a_2)}$
0	0	2/5
1	1	2/5
2	1	1/5

, $p(a_3)$ =

a_3	$f_{p(a_3)}$
0	2/5
1	3/5

, $p(a_4)$ =

a_4	$f_{p(a_4)}$
0	4/5
1	1/5

.

Figure 17: Marginal distributions $p(a_1, a_2), p(a_3)$, and $p(a_4)$ of the observed uniform distribution depicted in Figure 15.

Level 1, Pass 2: We now begin the second pass of the first level. That is, we again start with the empty undirected graph $G = (\mathcal{N}, E = \{\})$, and for this pass edge (a_1, a_3) is added. $G^1_{(a_1,a_3)}$ is shown in Figure 19. As in the first pass, respective marginal distributions are calculated as shown in Figure 20. The corresponding Markov distribution to this graph is constructed by:

$$p'(a_1, a_2, a_3, a_4) = p(a_1, a_3) \cdot p(a_2) \cdot p(a_4),$$

as shown in Figure 21. The corresponding entropy for this distribution is $\mathcal{H}(p(a_1, a_3) \cdot p(a_2) \cdot p(a_4)) = 1.2085761$. At this point, we have completed the second pass of the first level.

Level 1, Pass 3,...: For the rest of the passes of the first level, we show in Figure 22 the edge added to the empty undirected graph and the corresponding calculated entropy value.

$p(a_1, a_2) \cdot p(a_3) \cdot p(a_4) \;=$

a_1	a_2	a_3	a_4	$f_{p(a_1,a_2)\cdot p(a_3)\cdot p(a_4)}$
0	0	0	0	16/125
0	0	0	1	4/125
0	0	1	0	24/125
0	0	1	1	6/125
1	1	0	0	16/125
1	1	0	1	4/125
1	1	1	0	24/125
1	1	1	1	6/125
2	1	0	0	8/125
2	1	0	1	2/125
2	1	1	0	12/125
2	1	1	1	3/125

Figure 18: Level 1, Pass 1: Markov distribution $p'(a_1, a_2, a_3, a_4) = p(a_1, a_2) \cdot p(a_3) \cdot p(a_4)$ determined using the marginal distributions shown in Figure 17.

Since the entropy $\mathcal{H}(p(a_1, a_2) \cdot p(a_3) \cdot p(a_4))$ is the minimum entropy value for the first level, we choose this distribution and the corresponding graph $G^1_{(a_1,a_2)} = (\mathcal{N}, E = \{(a_1, a_2)\}$ for the starting distribution of level two.

Level 2, Pass 1: The first pass of second level adds the edge (a_1, a_3) to $G^1_{(a_1,a_2)}$ as depicted in Figure 23. With respect to $G^2_{(a_1,a_3)}$ we calculate, the required marginal distributions shown in Figure 24 using the distribution chosen in level one (depicted in Figure 18). Note that the marginal distribution $p(a_1)$ (not shown) can be determined from $p(a_1, a_2)$ or $p(a_1, a_3)$. Using these marginal distributions, we determine the Markov distribution:

$$p'(a_1, a_2, a_3, a_4) = p(a_1, a_2) \cdot p(a_1, a_3) \cdot p(a_4)/p(a_1).$$

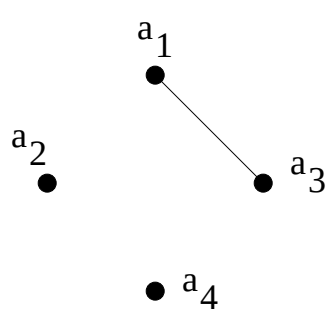

Figure 19: Level 1, Pass 2: The undirected graph $G^1_{(a_1,a_3)}$ constructed by adding the edge (a_1, a_3).

$p(a_1, a_3)$ =

a_1	a_3	$f_{p(a_1,a_3)}$
0	0	1/5
1	1	1/5
1	0	1/5
0	1	1/5
2	1	1/5

$p(a_2)$ =

a_2	$f_{p(a_2)}$
0	2/5
1	3/5

$p(a_4)$ =

a_4	$f_{p(a_4)}$
0	4/5
1	1/5

Figure 20: Level 1, Pass 2: marginal distributions $p(a_1, a_3), p(a_2)$, and $p(a_4)$ of the observed uniform distribution depicted in Figure 15.

The entropy for this distribution is $\mathcal{H}(p(a_1, a_2) \cdot p(a_1, a_3) \cdot p(a_4)/p(a_1)) = 0.09677528$.

Level 2, Pass 2,...: Each of the edges (a_1, a_4),(a_2, a_3),(a_2, a_4),(a_3, a_4) is added in turn to the chosen graph from level one $G^1_{(a_1,a_2)} = (\mathcal{N}, E = \{(a_1, a_2)\})$. The required marginal distributions are calculated and used to construct the Markov distribution p' for that pass. Using p', the corresponding $\mathcal{H}(p')$ is also calculated. For level two, the entropy $\mathcal{H}(p(a_1, a_2) \cdot p(a_1, a_4) \cdot p(a_3)/p(a_1))$ is the minimum entropy value. Thereby, we choose this distribution and the corresponding graph, $G^2_{(a_1,a_4)} = (\mathcal{N}, E = \{(a_1, a_2), (a_1, a_4)\})$ depicted in Figure 25 as the starting distribution for level three.

Level 3, Pass 1,...: Each of the edges (a_1, a_3),(a_2, a_3),(a_2, a_4),(a_3, a_4) is added in turn to the chosen graph from level two $G^2_{(a_1,a_4)} = (\mathcal{N}, E = \{(a_1, a_2), (a_1, a_4)\})$. As in the first two levels, the required marginal distributions are calculated and are used to construct the Markov distribution p' for that pass. Using p', the corresponding $\mathcal{H}(p')$ is also computed. For this level, the entropy $\mathcal{H}(p(a_1, a_2) \cdot p(a_1, a_3) \cdot p(a_1, a_4)/(p(a_1) \cdot p(a_1)))$ is the minimum entropy value. Thereby, we choose this distribution and the corresponding graph, $G^3_{(a_1,a_3)} = (\mathcal{N}, E = \{(a_1, a_2), (a_1, a_3), (a_1, a_4)\})$ depicted in Figure 26 as the starting distribution for level four.

Level 4, Pass 1,...: Each of the edges (a_2, a_3),(a_2, a_4),(a_3, a_4) is added in turn to the chosen graph from level three $G^3_{(a_1,a_3)} = (\mathcal{N}, E = \{(a_1, a_2), (a_1, a_3), (a_1, a_4)\})$. At each pass, the required marginal distributions are calculated and are used to construct the Markov distribution p' for that pass. Once p' is constructed, the corresponding $\mathcal{H}(p')$ is computed. At each pass i of level four, the entropy value $\mathcal{H}(G^4_{e_i}) = \mathcal{H}(G^3_{(a_1,a_3)})$. Clearly, the entropy decrement of level four is zero. This means that the graphical structure resulting, by adding one legal and possible edge to $G^3_{(a_1,a_3)}$, does not fit the observed data any better than $G^3_{(a_1,a_3)}$ alone. Thereby, the undirected graph $G^3_{(a_1,a_3)} = (\mathcal{N}, E = \{(a_1, a_2), (a_1, a_3), (a_1, a_4))$ is transformed into a hypertree as shown in Figure 27. Based on the concept of *separateness* (Hajek, Havranek & Jirousek, 1992), we infer the following MVDs which hold in the observed data: $a_1 \twoheadrightarrow a_2$, $a_1 \twoheadrightarrow a_3$, $a_1 \twoheadrightarrow a_4$. In this example, however, we fail to determine the MVD $a_4 \twoheadrightarrow a_3$. Note a similar concept of separateness is used in relational database theory (Fagin, Mendelzon & Ullman, 1982). □

$p(a_1, a_3) \cdot p(a_2) \cdot p(a_4)$

a_1	a_2	a_3	a_4	$f_{p(a_1,a_3)\cdot p(a_2)\cdot p(a_4)}$
0	0	0	0	8/125
0	0	0	1	2/125
0	0	1	0	12/125
0	0	1	1	3/125
1	1	0	0	8/125
1	1	0	1	2/125
1	1	1	0	12/125
1	1	1	1	3/125
1	0	0	0	8/125
1	0	0	1	2/125
1	0	1	0	12/125
1	0	1	1	3/125
0	1	0	0	8/125
0	1	0	1	2/125
0	1	1	0	12/125
0	1	1	1	3/125
2	1	0	0	8/125
2	1	0	1	2/125
2	1	1	0	12/125
2	1	1	1	3/125

Figure 21: Level 1, Pass 2: Markov distribution $p'(a_1, a_2, a_3, a_4) = p(a_1, a_3) \cdot p(a_2) \cdot p(a_4)$ determined using the marginal distributions shown in Figure 20.

As already stated, the problem of finding all MVDs which hold is NP-complete (Bouckaert, 1994). We should perhaps comment here why the MVD $a_4 \twoheadrightarrow a_3$ is not found. Algorithm 1 derives an acyclic hypergraph (i.e., a hypertree) using the observed data. Beeri et al. (Beeri et al., 1983) proved that an acyclic join dependency is equivalent to a *conflict-free* set of MVDs. It can be verified, however, that the set of MVDs $\{ a_1 \twoheadrightarrow a_2$, $a_1 \twoheadrightarrow a_3$, $a_1 \twoheadrightarrow a_4$, $a_4 \twoheadrightarrow a_3 \}$, which hold in the observed data in Example 5, are *not* conflict-free. This means that this set of MVDs cannot be represented as an acyclic join dependency (i.e., a uniform Markov distribution). We therefore fail to learn all the MVDs in this example. It should be noted that the output of our algorithm is a desirable acyclic database schema which is known to possess many desirable properties (Beeri et al., 1983).

We would like to highlight a subtle difference between Algorithm 1 and the simulation of Algorithm 1 in Example 5. In the simulation of Algorithm 1 in Example 5 each level i would start with an initial distribution, say p. For each pass j of level i, respective marginal distributions would be computed. These marginal distributions would then be multiplied to create a new distribution p'. The entropy $\mathcal{H}(p')$ would then be computed using equation (15). However, in the definition of Algorithm 1 we state that the entropy decrement is computed *locally*. This means that the entire distribution need not be constructed in order to compute

$EdgeAdded$	$p'(a_1, a_2, a_3, a_4) =$	$\mathcal{H}(p')$
(a_1, a_2)	$p((a_1, a_2) \cdot p(a_3) \cdot p(a_4))$	0.9673925
(a_1, a_3)	$p((a_1, a_3) \cdot p(a_2) \cdot p(a_4))$	1.2085761
(a_1, a_4)	$p((a_1, a_4) \cdot p(a_2) \cdot p(a_3))$	1.0870448
(a_2, a_3)	$p((a_2, a_3) \cdot p(a_1) \cdot p(a_4))$	1.2540246
(a_2, a_4)	$p((a_2, a_4) \cdot p(a_1) \cdot p(a_3))$	1.2085765
(a_3, a_4)	$p((a_3, a_4) \cdot p(a_1) \cdot p(a_2))$	1.2085765

Figure 22: Level 1: summary including the edge added, representation of the corresponding Markov distribution p', and calculated entropy $\mathcal{H}(p')$.

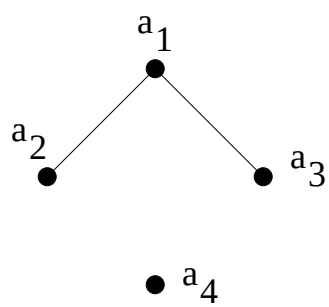

Figure 23: Level 2, Pass 1: The undirected graph $G^2_{(a_1,a_3)}$ constructed by adding the edge (a_1, a_3) to $G^1_{(a_1,a_2)}$.

the corresponding entropy. More specifically, given the initial distribution for level i, for each pass j, only the required marginal distributions need be computed. The entropy for the joint distribution can be calculated without forming the product of the marginal distributions. To illustrate the concept of computing the entropy decrement locally, consider the following situation which uses the same attributes as in Example 5. The entropy of the initial empty graph can be computed using the formula

$$\mathcal{H}(p(a_1)p(a_2)p(a_3)p(a_4)) = \mathcal{H}(p(a_1)) + \mathcal{H}(p(a_2)) + \mathcal{H}(p(a_3)) + \mathcal{H}(p(a_4)). \tag{23}$$

After adding the edge (a_1, a_2) the corresponding entropy for this new distribution can be computed by

$$\mathcal{H}(p(a_1, a_2)p(a_3)p(a_4)) = \mathcal{H}(p(a_1, a_2)) + \mathcal{H}(p(a_3)) + \mathcal{H}(p(a_4)). \tag{24}$$

The entropy decrement between these two distributions can now be computed with

$$\begin{aligned} [(entropy-decrement-level) &- (entropy-decrement-pass)] \\ &= \mathcal{H}(p(a_1)p(a_2)p(a_3)p(a_4)) - \mathcal{H}(p(a_1, a_2)p(a_3)p(a_4)) \\ &= \mathcal{H}(p(a_1)) + \mathcal{H}(p(a_2)) - \mathcal{H}(p(a_1, a_2)). \end{aligned} \tag{25}$$

$p(a_1, a_2)$ =

a_1	a_2	$f_{p(a_1,a_2)}$
0	0	10/25
1	1	10/25
2	1	5/25

$p(a_1, a_3)$ =

a_1	a_3	$f_{p(a_1,a_3)}$
0	0	4/25
0	1	6/25
1	0	4/25
1	1	6/25
2	0	2/25
2	1	3/25

$p(a_4)$ =

a_4	$f_{p(a_4)}$
0	20/25
0	5/25

Figure 24: Level 2, Pass 1: marginal distributions $p(a_1, a_2), p(a_1, a_3)$ and $p(a_4)$ of the distribution depicted in Figure 18.

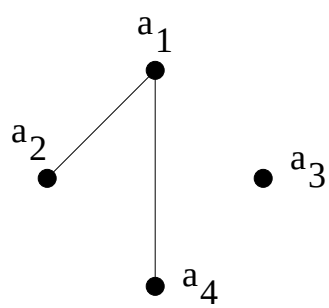

Figure 25: Level 2: The corresponding undirected graph $G^2_{(a_1,a_4)}$ chosen from level two.

Thereby, for calculating the effect of adding edge (a_1, a_2) to the undirected graph, only the relevant entropies need be computed. In this situation, $\mathcal{H}(p(a_3))$ and $\mathcal{H}(p(a_4))$ do not have to be calculated as they cancel each other in equation (25). Intuitively, the concept of calculating the entropy for a marginal distribution is shown in equation (15), while computing the entropy for a distribution using the entropies of marginal distributions is shown in equation (18). The associated problems with computing the entropy locally are described in (Xiang, Wong & Cercone, 1995). Let us now return to discussing the experiments in this paper.

The observed data used in the experiments were obtained in one of two fashions. The first fashion was to use brute force to manually create a large collection of tuples. Informally, this was accomplished by initially designing copious, small individual relations. The next task was simply to join all these relations together creating one large repository of data. Formally, we created $r_i, 1 \leq i \leq n$, with the schema for each r_i denoted by R_i. The repository of tuples, denoted r, over schema $R = R_1 \cup R_2 \cup \ldots \cup R_n$, was created by the computation $r = r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n \ = \bowtie r_i, 1 \leq i \leq n$. It has been shown (Maier, 1983) that r, constructed in this fashion, will satisfy the MVD $X \twoheadrightarrow Y$, where $X = R_1 \cap R_2$ and $Y = R_1 - X$. Extending this notion, r encodes at least $n - 1$ MVDs. This collection of known MVDs serves as our control set of MVDs. We can thereby compare the output of our system with the control set of MVDs.

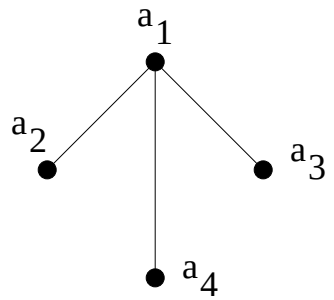

Figure 26: Level 3: The corresponding undirected graph $G^3_{(a_1,a_3)}$ chosen from level three.

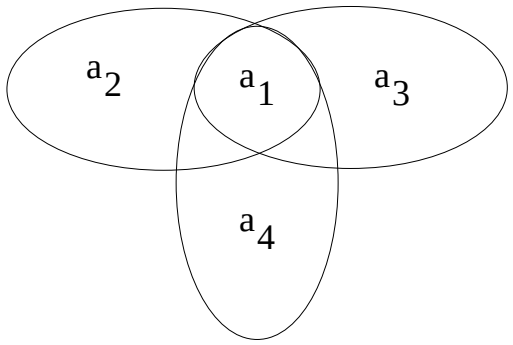

Figure 27: The hypertree, corresponding to the undirected graph $G^3_{(a_1,a_3)}$, constructed as output by Algorithm 1 using the observed data in Figure 15 as input.

The second approach to constructing the observed data involved modifying sample data used in the experiments for our algorithm for determining probabilistic conditional independencies. Essentially, this sample data for probabilistic reasoning consisted of a large database which contained duplicate tuples. These large databases were originally constructed from Bayesian networks of famous probabilistic experiments, namely the Alarm experiment (Beinlich et al., 1989) and the Fire experiment (Poole & Neufeld, 1988). As no duplicates are allowed in relational theory, the only task remaining was to remove all duplicates from the database. All probabilistic conditional dependencies which were found in our probabilistic inference experiments should be found in our relational theory experiments. This is due to the fact that MVD is a necessary condition for probabilistic conditional independence. However, since MVD is not a sufficient condition for probabilistic conditional independence, additional MVDs may be found in our relational experiments. Thereby, in this approach for constructing the observed data, the control set is the set of probabilistic conditional independencies which we know hold in the corresponding probabilistic database.

n	$n/125$	$\log n/125$	$n/125 \log n/125$
2	0.016	-1.79588	-0.028734
3	0.024	-1.6197888	-0.0388749
4	0.032	-1.49485	-0.0478352
6	0.048	-1.3187588	-0.0633004
8	0.064	-1.19382	-0.0764044
12	0.096	-1.0177288	-0.0977019
16	0.128	-0.89279	-0.1142771
24	0.192	-0.7166987	-0.1376061

Figure 28: Numerical calculations used in Example 5.

7 Algorithm Analysis and Experimental Results

For simplicity we have presented our multi-link algorithm as a single-link algorithm. The algorithm analysis, however, will be performed on the more general multi-link technique. To reduce the complexity of a multi-link lookahead search, we make a spareness assumption. A parameter η, specified by the user, will bound the size of the clique searched for during the execution of Algorithm 1. We now analyze the worst case time complexity of the algorithm.

Testing the chordality of *G-pass* can be performed in $O(\mid \mathcal{N} \mid)$ time (Golumbic, 1980). A hypertree (junction tree) can be computed by a maximal spanning tree algorithm (Jensen, 1988). A maximal spanning tree of a graph with v nodes and e links can be computed in $O((v+e) \log v)$ time (Manber, 1989). Since a complete graph has $O(v^2)$ links, a maximal spanning tree can be computed in $O(v^2 \log v)$ time. Equivalently, computation of a hypertree of a chordal graph with k nodes with v cliques takes $O(v^2 \log v)$ time. Since $v \leq k$, computation of a hypertree of a chordal graph with k nodes takes $O(k^2 \log k)$ time. In computing *entropy-decrement-pass*, we need to compute F and *F-pass* from the corresponding chordal subgraphs. Each of them contains no more than 2η attributes, where η is the maximum allowable size of a clique. Therefore, we can compute F and *F-pass* in $O(\eta^2 \log \eta)$ time.

Let n be the number of cases in the database. We can extract the distribution p' on the 2η attributes from the database directly in $O(n)$ time. The projected distribution on F and *F-pass* can be computed by marginalizing p' to cliques and multiplying clique distributions, which takes $O(\eta 2^\eta)$ time. The computation of *entropy-decrement-pass* from the projected distributions can be performed in $O(2^\eta)$ time. The complexity of each step is then $O(\mid \mathcal{N} \mid +n + \eta(\eta \log \eta + 2^\eta))$. Since n is much larger than $\mid \mathcal{N} \mid$, the complexity of each search step is $O(n + \eta(\eta \log \eta + 2^\eta))$.

Let κ be the number of lookahead links. The algorithm repeats for $O(\kappa)$ levels. Each level contains $O(\mid \mathcal{N} \mid^2)$ passes. Each pass has $C(C(\mid \mathcal{N} \mid, 2), \kappa) = O(\mid \mathcal{N} \mid^{2\kappa})$ steps. Hence, the algorithm has $O(\kappa \mid \mathcal{N} \mid^{2\kappa})$ search steps. The overall complexity of the algorithm is then $O(\kappa \mid \mathcal{N} \mid^{2\kappa} (n + \eta(\eta \log \eta + 2^\eta))$. This computation is feasible if κ and η are small.

The main objective of our experiments is to check how well our method outlined in

Section 6 is able to discover the MVDs (probabilistic conditional independencies) which hold in observed data. We now present our experimental results as shown in Figure 29, where $\mathcal{N}$ is the number of attributes in the observed data, n is the number of (unique) tuples in the observed data, $Known$ is the number of MVDs we know hold in the observed data, and $Discovered$ is the number of discovered MVDs in the observed data.

$\mathcal{N}$	n	$Known$	$Discovered$
4	5	4	3
6	13	1	0
40	29	9	8
20	176	15	14
26	260	12	23
30	264	10	31
10	276	3	7
20	4428	6	16
37	5046	30	25

Figure 29: Experimental results consisting of 9 examples of various sizes.

We can see from Figure 29 that in four of the examples our algorithm discovered all the known MVDs plus many more which held. In four of the other five examples, our algorithm only failed to discover one MVD. This clearly demonstrates that our proposed method is very effective in learning MVDs from observed data.

8 Conclusion

Data dependencies are used in database schema design to enforce the correctness of a database as well as to reduce redundant data. These dependencies are usually determined from the semantics of the attributes and are then enforced upon the relations. Very often, however, we do not have the necessary semantic information to determine the dependency relationships amongst a given set of attributes. Even if some dependencies are known, the database designer may still want to know whether other hidden dependencies exist. In these situations, we therefore need an inference mechanism to *discover* the relevant dependencies from observed data for designing an appropriate database schema.

In this paper, we have suggested a bottom-up procedure for mining MVDs in observed data without knowing à priori the dependencies amongst the attributes. Our technique is an adaption of the method we developed for learning probabilistic conditional independencies. This is possible because the notion of MVD is equivalent to that of probabilistic conditional independence in a uniform distribution. To reduce the complexity of computation, a single-link search technique was used in the minimization procedure as described in Algorithm 1. Our experimental results indicate that this simplified approach is adequate in practical situations. In general, however, one may have to adopt a multi-link search method in order

to discover certain types of MVDs. It should be noted that the proposed method does not discover *all* the MVDs that can be logically implied by the observed data. It has been shown that discovering all the probabilistic conditional independencies in a probability distribution is a NP-hard problem (Bouckaert, 1994).

The results of our experiments are rather encouraging. They indicate that our method is both effective and efficient. It is understood that the computation would become intractable if the number of attributes involved is very large.

We have implemented our learning algorithm as a prototype system. Given observed data as input, our system will produce as output an acyclic database schema. Thus, our system can be used as a tool for automated database schema design. As already mentioned, the output schema always satisfies an acyclic join dependency which has been shown to possess a number of desirable properties (Beeri et al., 1983) in database applications.

References

Beeri, C., Fagin, R. & Howard, J. (1977). A complete axiomatization for functional and multivalued dependencies in database relations. *Proceedings of the ACM SIGMOD Conference*, 47-61.

Beeri, C., Fagin, R. Maier, D. & Yannakakis, M. (1983). On the desirability of acyclic database schemes. *Association for Computing Machinery*, 30, 479–513.

Beinlich, I., Suermondt, H., Chavez, R. & Cooper, G. (1989). The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. *Technical Report KSL-88-84*, Knowledge Systems Lab, Medical Computer Science, Stanford University.

Bouckaert, R. (1994). Properties of Bayesian belief network learning algorithms. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 102-109.

Cooper, G.F., & Herskovits, E.H., (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9, 309-347.

Dechter, R. (1990). Decomposing a relation into a tree of binary relations. *Computer and System Sciences*, 41, 2-24.

Delobel, C. (1978). Normalization and hierarchical dependencies in the relational data model. *ACM Transactions on Database Systems*, 3(2-3), 201-222.

Fagin, R. (1977). Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems*, 2(3), 262-278.

Fagin, R., Mendelzon, A.O. & Ullman, J.D. (1982). A simplified universal relation assumption and its properties. *ACM Transactions on Database Systems*, 7, 343-360.

Flach, P. (1990). Inductive characterization of database relations. *Methodologies for Intelligent Systems*, 5, Z.W. Ras, M. Zemankowa, M.L. Emrich (eds.), North-Holland, Amsterdam, 371-378.

Golumbic, M. (1980). *Algorithmic Graph Theory and Perfect Graphs.* Academic Press.

Hajek, P., Havranek, T. & Jirousek, R. (1992). *Uncertain Information Processing in Expert Systems.* CRC Press.

Heckerman, D., Geiger, D. & Chickering, D.M. (1995). Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20, 197-243.

Herskovits, E.H., & Cooper, G.F. (1990). Kutato: an entropy-driven system for construction of probabilistic expert systems from database. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 54-62.

Hill, J. (1993). Comment. *Statistical Science*, 8, 3, 258-261.

Jensen, F. (1988). Junction tree and decomposable hypergraphs. *Technical report, JUDEX*, Aalborg, Denmark.

Jensen, F. (1996). *An Introduction to Bayesian Networks.* UCL Press.

Kullback, S., & Leibler, R. (1951). Information and sufficiency. *Annals of Mathematical Statistics*, 22, 79-86.

Lam, W., & Bacchus, F. (1994). Learning Bayesian networks: an approach based on the mdl principle. *Computational Intelligence*, (10)3, 269-293.

Lauritzen, S., & Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Royal Statistical Society*, B, 50, 2, 157-224.

Lee, T. (1983). An algebraic theory of relational databases. *The Bell System Technical Journal*, 62, 10, 3159-3204.

Maier, D. (1983). *The Theory of Relational Databases.* Computer Science Press.

Manber, U. (1989). *Introduction to Algorithms: a Creative Approach.* Addison-Wesley.

Neapolitan, R. (1990). *Probabilistic Reasoning in Expert Systems.* John Wiley & sons.

Poole, P., & Neufeld, E. (1988). Sound Probabilistic Inference in Prolog: an executable specification of influence diagrams. *Proceedings of the I Simposium Internacional De Inteligencia Artificial.*

Pearl, P., & Verma, T. (1987). The logic of representing dependencies by directed graphs. *Proceedings of the AAAI87 Sixth National Conference on Artificial Intelligence*, 1, 374-379.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Rebane, G. (1987). The recovery of causal play-trees from statistical data. *Proceedings of Workshop on Uncertainty in Artificial Intelligence*, 222-228.

Savnik, I, & Flach, P. (1993). Bottom-up induction of functional dependencies from relations. *Proceedings of the AAAI-93 Workshop: Knowledge Discovery in Databases*, 174-185.

Shafer, G. (1991). An axiomatic study of computation in hypertrees. *School of Business Working Paper Series*, 232, University of Kansas, Lawrence.

Spirtes, P., & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1), 62-73.

Xiang, Y., Wong, S.K.M. & Cercone, N. (1995). A 'microscopic' study of minimum entropy search in learning decomposable markov networks. Accepted by *Machine Learning*.

Xiang, Y., Wong, S.K.M. & Cercone, N. (1996). Critical remarks on single link search in learning belief networks. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 564-571.

Wong, S.K.M. (1994a). An extended relational data model for probabilistic reasoning. Accepted by *Journal of Intelligent Information Systems*.

Wong, S.K.M., & Xiang, Y. (1994b). Construction of a markov network from data for probabilistic inference. *Proceedings of the 3rd International Workshop on Rough Sets and Soft Computing*, 562-569.

Wong, S.K.M., Xiang, Y. & Nie, X. (1994c). Representation of Bayesian networks as relational databases. *Proceedings of the Fifth International Conference Information Processing and Management of Uncertainty in Knowledge-based systems*, 159-165.

Wong, S.K.M., Butz, C.J. & Xiang, Y. (1995). A method for implementing a probabilistic model as a relational database. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 556-564.

Wong, S.K.M. (1996). Testing implication of probabilistic dependencies. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 545-563.