JOURNAL OF COMPUTER AND SYSTEM SCIENCES 41, 2–24 (1990)

Decomposing a Relation into a Tree of Binary Relations*

RINA DECHTER†

Computer Science Department, Cognitive Systems Laboratory, University of California, Los Angeles, California 90024

Received July 8, 1987; revised February 26, 1989

We present an efficient algorithm for decomposing an n-ary relation into a tree of binary relations and provide a simple test for checking whether or not the tree formed by the algorithm represents the relation precisely. If a tree decomposition exists, the algorithm is guaranteed to find one. Otherwise, the tree generated by the algorithm can be used as an approximation of the relation. The method can be extended to decomposing a relation into an acyclic database of bounded relation arity. The unique feature of the algorithm presented in this paper is that it does not a priori assume any dependencies in the initial relation, but rather derives such dependencies from the bare relation instance. © 1990 Academic Press, Inc.

1. INTRODUCTION

The primary use of functional dependencies and multivalued dependencies in relational databases is to guide the decomposition of a relation scheme into a database scheme consisting of smaller relations that satisfy the join-dependency property, i.e., the reconstruction of the whole relation from its components by the natural join operation is lossless [11]. The purpose of such decomposition is to attenuate data redundancy and enhance data reliability. Query processing, on the other hand, which may sometimes require the reconstruction of the whole relation, becomes more expensive.

Some of the shortcomings of decomposition are avoided if the relation is decomposed into a *tree of binary relations.* Being a special case of acyclic database schemes, such trees inherit the latter's desirable property [2], e.g., efficient query processing, and, at the same time, require minimum storage space.

In this paper we present a greedy algorithm which is guaranteed to produce a lossless decomposition of a relation into a tree of binary relations when such decomposition exists, along with a simple test for determining whether the tree produced by the algorithm is indeed a lossless decomposition. If the relation is not *tree-decomposable*, then we can use the tree decomposition of the relation, together with that of its complement, as an approximate representation. Alternatively, the

* This work was supported in part by the National Science Foundation, Grant IRI–8815522 and by Air Force Office of Scientific Research, Grant AFOSR 88 0177.

† Current affiliation, Computer Science Department, Technion, Haifa, Israel.

0022-0000/90 $3.00

decomposition scheme can be extended to form acyclic database instances where the arity of each subrelation is greater than two.

Since the method operates on a single database instance, and since the decomposability of a relation may change as a result of *add* and *delete* operations, the practicality of the method we propose is limited to databases requiring infrequent updates. This is the case, for example, when databases are used for describing natural phenomena such as relations between diseases and their characteristic manifestations or for describing complex physical systems and their possible modes of failure.

Another potential use of the algorithm is to assist a database designer by suggesting candidate multivalued dependencies [6] (MVDs) that fit an initial relation instance. This set of MVDs can be inspected by the designer to select those he may wish to impose on all instances.

The method was motivated by an algorithm developed by Chow and Liu [3] for approximating discrete probability distributions with *dependence tree*. Related work is presented in [13–15].

2. Definitions and Preliminaries

A data base instance can be associated with a hypergraph where the nodes represent the attributes and the hyperarcs are subsets of attributes appearing in the same relation. When all relations in the instance are binary, i.e., containing only two attributes, the hypergraph reduces to an ordinary graph, called a *constraint graph*. In the special case where the constraint graph is a tree, a useful result states that the database can be processed very efficiently [2], for instance, one tuple in the relation instance can be generated in $O(nk^2)$, where n is the number of attributes and k is their domain size [4].

Consider, for example, the relation on FLIGHT, DAY-OF-WEEK, and PLANE-TYPE presented in Fig. 1 (from [11]). This relation can be decomposed losslessly into relations on the pairs (FLIGHT, DAY-OF-WEEK) and (FLIGHT, PLANE-TYPE). The associated graph is given in Fig. 2.

Let ρ denote an n-ary relation over the set of attributes $U = \{X_1, ..., X_n\}$, i.e., a subset of the Cartesian product $\text{Dom}(X_1) \times , ..., \times \text{Dom}(X_n)$, where $\text{Dom}(X_i)$ is the

FLIGHT	DAY-OF-WEEK	PLANE-TYPE
106	Monday	747
106	Thursday	747
106	Monday	1011
106	Thursday	1011
204	Wednesday	707
204	Wednesday	727

FIG. 1. The relation (FLIGHT, DAY-OF-WEEK, PLANE-TYPE).

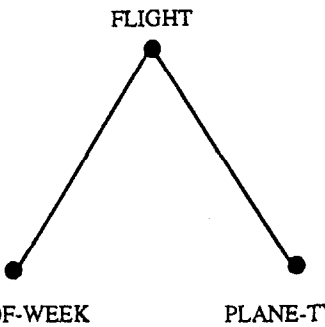


FIG. 2. The graph associated with (FLIGHT, DAY-OF-WEEK), (FLIGHT, PLANE-TYPE).

set of values of attribute X_i. Let ρ_S denote the projection of ρ on a subset S of attributes, namely, ρ_S is obtained from ρ by striking out columns corresponding to attributes $U-S$ and removing duplicate tuples in the columns that remain. Given an attribute A and an element a of its domain, the *restriction of* ρ *to* $A=a$, denoted by $\rho^{r(A=a)}$, is the n-ary relation containing all n-tuples in ρ having value a for A. Similarly, the restriction of ρ to a subtuple $t=(X_{i1}=x_{i1}, ..., X_{ir}=x_{ir})$, denoted by $\rho^{r(t)}$, consists of all the n-tuples of ρ that match t for the corresponding attributes.

The property of a relation that enables its lossless decomposition into two smaller relations is what we call *conditional independence.*

DEFINITION. Let S_1, S_2, S_3 be subsets of U. S_1 and S_2 are conditionally independent given S_3, denoted by $\langle S_1 | S_3 | S_2 \rangle$, if for every combination of values for attributes in S_3, denoted by $S_3 = s_3$, we have

$$L^{r(S_3=s_3)} = (L_{S_1S_3})^{r(S_3=s_3)} \bowtie (L_{S_2S_3})^{r(S_3=s_3)},$$

where

$$L = \rho_{S_1S_2S_3}.$$

If $\langle S_1 | S_3 | S_2 \rangle$ holds in a relation ρ, and $U = S_1S_2S_3$, then ρ can be decomposed losslessly into the database scheme S_1S_3 and S_2S_3. For instance, in the relation of Fig. 1, the attribute DAY-OF-WEEK is conditionally independent of PLANE-TYPE given FLIGHT. Conditional independence parallels the notion of embedded-multi-valued-dependencies (EMVDs) [6]. That is, $\langle S_1 | S_3 | S_2 \rangle$ iff $S_3 \twoheadrightarrow S_2$ in the projection of ρ on $S_1S_2S_3$. We will use the terms conditional independence and EMVDs interchangeably.

A constraint graph, representing a relation, explicates some of its conditional independencies. In a graph, S_2 is said to *separate* S_1 from S_3 if by removing the nodes in S_2, nodes in S_1 are disconnected from nodes in S_3. Every *separation* in the constraint graph corresponds to a conditional independence, i.e., if a subset S_1 separates (in the constraint graph) subset S_2 from S_3 then $\langle S_2 | S_1 | S_3 \rangle$. However, there may be conditional independencies in the relation which are not manifested

in the constraint graph. For a detailed discussion of conditional independencies and their graphical representation see [17, 18].

The following notations will be used throughout:

$n_\rho(X_i = x_i) \triangleq$ number of n-tuples in ρ for which $X_i = x_i$

$n_\rho(X_i = x_i, X_j = x_j) \triangleq$ number of n-tuples in ρ for which $X_i = x_i$ and $X_j = x_j$. In general,

$n_\rho(X_{i_1} = x_{i_1}, ..., X_{i_t} = x_{i_t}) \triangleq$ number of n-tuple in ρ for which $X_{i_1} = x_{i_1}, ..., X_{i_t} = x_{i_t}$.

Let S be any subset of U. We define, $n_S(X_i = x_i) \triangleq$ number of $|S|$-tuples in the projection ρ_S for which $X_i = x_i$, namely, $n_S(\cdot)$ is used as a shorthand for $n_{\rho_S}(\cdot)$. In general, $n_S(X_{i_1} = x_{i_1}, ..., X_{i_t} = x_{i_t}) \triangleq$ number of $|S|$-tuple in ρ_S for which $X_{i_1} = x_{i_1}, ..., X_{i_t} = x_{i_t}$.

When the referenced relation is the global relation, ρ, we may simplify $n_\rho(\cdot)$ to $n(\cdot)$, and, when there is no confusion regarding the identity of the attributes, we will simplify $n(X_{i_1} = x_{i_1}, ..., X_{i_t} = x_{i_t})$ to $n(x_{i_1}, ..., x_{i_t})$.

3. Theoretical Grounds for Tree Decomposition

The following is a demonstration of the algorithm which will be developed in the sequel. Consider the relation over the binary attributes $\{A, B, C, D, E\}$ given in Fig. 3. The first step of the algorithm computes the quantities $\{n(X_i = x_i)\}$ and $\{n(X_i = x_i, X_j = x_j)\}$ for all attributes and their values. Obtaining:

$$n(A = 0) = 8, \qquad n(B = 1) = 6, \qquad n(B = 0) = 2,$$

$$n(B = 0, C = 1) = 2, \qquad n(B = 1, C = 1) = 3, \qquad \text{etc.}$$

Next, for each pair of attributes (X_i, X_j) we compute the weights $m(X_i, X_j)$ according to the formula given in Eq. (32) of Section 4,

$$m(A, B) = m(A, C) = m(A, D) = m(A, E) = -16.63,$$

$$m(B, C) = -13.97, \qquad m(B, D) = -15.95, \qquad m(B, E) = -16.55,$$

$$m(C, D) = -16.55, \qquad m(C, E) = -17.13, \qquad m(D, E) = -15.50.$$

A	B	C	D	E
0	0	1	1	1
0	0	1	1	0
0	1	1	1	1
0	1	1	1	0
0	1	1	0	1
0	1	0	1	1
0	1	0	1	0
0	1	0	0	1

Figure 3

Finally, using the maximum-weight spanning-tree algorithm on these weighted arcs, the tree shown in Fig. 4 is produced. The relations associated with the arcs of the tree are the projections of the global relation on pairs of connected attributes. For instance, the relation associated with attributes D and B is ρ_{DB}. The tree generated in this example, together with its associated database (see Fig. 4), represents the original relation, in the sense that it provides a lossless decomposition. The relation can be efficiently recovered by performing the natural join operation in the partial order dictated by the tree going from leaves to the root.

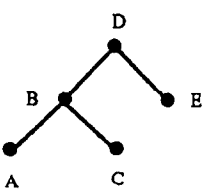
The tree structure also explicates several conditional independencies (i.e., EMVDs). For instance, since attribute D separates attributes B, A, C, from E, the conditional independence, $\langle BAC | D | E \rangle$ holds. In trees every attribute separates the graph into two subsets, and each separation represents a genuine EMVD in the relation.

The following paragraphs contain the justification for this algorithm. It is partially based on a result by Chow and Liu [3] concerning an optimal approximation of a probability distribution by a tree-dependent distribution. We shall first summarize Chow and Liu's result and then establish a relationship between relations and probability distributions, from which a similar algorithm for finding tree decompositions for relations will emerge.

Let $P(x)$ be a probability distribution of n variables, $X_1, X_2, ..., X_n$, and let T be a tree connecting the n variables. To our convenience we use both the n-tuple $x_1, x_2, ..., x_n$ and the symbol x to denote the set of n propositions $X_1 = x_1$, $X_2 = x_2, ..., X_n = x_n$. The tree-dependent distribution associated with P and T, denoted by P^T, is defined by the product form

$$P^T(x_1, ,..., x_n) = \prod_{i=1}^{n} P(x_i | x_{j(i)}), \tag{1}$$

where $X_{j(i)}$ is the variable designated as the parent of X_i in some root-down orientation of the tree. The root X_1 can be chosen arbitrarily and, having no parents, is characterized by the prior probability $P(x_1 | x_0) = P(x_1)$ (see Fig. 5). Chow and Liu asked the following question: Given a probability distribution P, what is the tree-dependent distribution P^T that best approximates P? In other words, among all the spanning trees that one can draw on n variables, each yielding a product form P^T



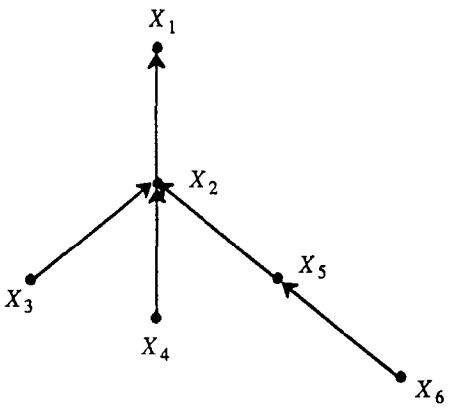
D	B
1	0
1	1
0	1

D	E
1	1
1	0
0	1

A	B
0	0
0	1

B	C
0	1
1	1
1	0

FIGURE 4



$$P^T(x) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2)P(x_4 \mid x_2)P(x_5 \mid x_2)P(x_6 \mid x_5)$$

FIG. 5. A tree-dependent distribution.

(see (1)), which P^T will be closest to P? As a distance criterion between two distributions, P and P^T, they chose the cross-entropy measure [8, 19]

$$D(P, P^T) = \sum_x P(x) \log \frac{P(x)}{P^T(x)}. \tag{2}$$

This measure is non-negative and attains the value zero iff P^T coincides with P. Surprisingly, they found a simple solution:

THEOREM 1 [3]. The distance measure (2) is minimized when T is any maximum weight spanning tree (MWST), where the weights of all arcs (X_i, X_j) are defined by the mutual information measure

$$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)\,P(x_j)} \geqslant 0. \tag{3}$$

By virtue of this result, the minimization can be solved, without exhaustively considering all possible trees, by the efficient MWST algorithm [5].

We now address the question whether it is possible to extend Chow and Liu's result from probabilities to relations in the following intuitive way. Suppose that a relation ρ is associated with a uniform probability distribution which accords equal non-zero probabilities to all tuples in the relation and to those tuples only. Does the existence of an exact tree-dependent distribution for this uniform distribution imply that the relation is tree decomposable? Moreover, does a tree-decomposable relation necessarily have a tree-dependent uniform probability distribution? Are they decomposable by the same tree? And if so, can we use the method to find an optimal tree-decomposition approximation to a given relation? We will formalize

these questions and show that the answer is in the affirmative for exact decomposition but remains unsettled for approximate decompositions.

Let ρ be an n-ary relation on attributes $X_1, ..., X_n$. We define by $PP(\rho)$ the set of all probability distributions that associate non-zero probabilities with tuples in ρ, and with these tuples only. Namely,

$$PP(\rho) = \{P \mid P(x) > 0 \ \forall x \in \rho \quad \text{and} \quad P(x) = 0 \ \forall x \notin \rho\}. \tag{4}$$

An arbitrary member of $PP(\rho)$ will be denoted by P_ρ. In particular, the uniform probability distribution, denoted by U_ρ, is a member of $PP(\rho)$. It is defined by

$$U_\rho(x) = \frac{1}{|\rho|}, \qquad \forall x \in \rho, \tag{5}$$

where $|\rho|$ is the cardinality of ρ.

Given a probability distribution on n variables (attributes) we define the relation rel(P), associated with P, as the set of all tuples that have non-zero probabilities, namely,

$$\text{rel}(P) = \{x \mid P(x) > 0\}. \tag{6}$$

Clearly,

$$\rho = \text{rel}(P_\rho), \qquad \forall P_\rho \in PP(\rho). \tag{7}$$

For a probability distribution P and a tree T, the tree-dependent distribution P^T is defined in (1). Similarly, for a relation ρ and a tree T we define the tree-dependent relation ρ^T as the relation generated by projecting ρ on pairs of attributes connected in the tree and then joining these binary relations. Formally,

$$\rho^T = \bowtie_{(X_i, X_j) \in T} \rho_{X_i X_j}. \tag{8}$$

Next, we show that if P is tree dependent along a tree T, then rel(P) has a lossless decomposition by the same tree.

THEOREM 2. *If* $P = P^T$ *then* $(\text{rel}(P) = \text{rel}(P))^T$.

Proof. For every relation ρ and a tree T it is always the case that $\rho^T \supseteq \rho$. In particular, $(\text{rel}(P))^T \supseteq \text{rel}(P)$. We have to show that if a tuple is not a member of rel(P) it is also not a member of $(\text{rel}(P))^T$. Suppose that $x \notin \text{rel}(P)$. From the definition of rel(P), it follows that $P(x) = 0$, and since T supports an exact tree-dependent distribution of P, it also follows that $P^T(x) = 0$. This last equality translates to

$$P^T(x) = \prod_{(X_i, X_{j(i)}) \in T} P(x_i \mid x_{j(i)}) = 0, \tag{9}$$

which implies that at least one component of the product must be zero. Assume that $P(x_{i_0} | x_{j(i_0)}) = 0$, then, also $P(x_{i_0}, x_{j(i_0)}) = 0$. Since $\forall x$, $P(x) \geq 0$ and, since

$$0 = P(x_{i_0}, x_{j(i_0)}) = \sum_{\{x \mid X_{i_0} = x_{i_0}, X_{j(i_0)} = x_{j(i_0)}\}} P(x), \tag{10}$$

all tuples x having $X_{i_0} = x_{i_0}$, $X_{j(i_0)} = x_{j(i_0)}$ must have $P(x) = 0$, and, therefore, do not appear in rel(P). Consequently, the projection of rel(P) on these two attributes cannot contain the tuple $(x_{i_0}, x_{j(i_0)})$. Thus

$$(x_{i_0}, X_{j(i_0)}) \notin (\text{rel}(P))_{X_{i_0} X_{j(i_0)}}. \tag{11}$$

Since $(X_{i_0}, X_{j(i_0)})$ is an arc in the tree T, it is impossible that x will be a member of $(\text{rel}(P))^T$. ∎

Theorem 2 implies that for a given relation ρ, if U_ρ is an exact tree-dependent distribution, U^T, then the tree T supports a lossless decomposition of ρ and the MWST algorithm on U_ρ will generate such a tree. In the continuation of this section we show that a partial converse of Theorem 2 is also true, namely, that if ρ can be decomposed losslessly into some tree T, then its associated uniform distribution, U_ρ, is indeed tree dependent along T. Therefore, the MWST algorithm *is guaranteed* to produce a tree decomposition to ρ if such a decomposition exists. It is not true, however, that every probability P_ρ is tree-dependent when ρ is tree decomposable. The main result is stated in Theorem 4 below. But first, we need the following theorem which establishes a product form for tuples of a tree decomposed relation.

THEOREM 3. *If ρ is representable by a tree, T, namely $\rho = \rho^T$, then $\forall x \in \rho$,*

$$n(x_1) \prod_{(X_i, X_{j(i)}) \in T} \frac{n(x_i, x_{j(i)})}{n(x_{j(i)})} = 1. \tag{12}$$

The following subsections establish the proof of Theorem 3 via several lemmas.

We say that a (constraint) graph *represents* the relation ρ, if ρ can be losslessly decomposed into the binary relations corresponding to the arcs of the graph. If, in a constraint graph, an attribute X_i separates two subsets of attributes, S_1 and S_2, each containing X_i, then, in the joined relation each value of X_i which appears in ρ_{S_1} will be duplicated as many times as it appears in ρ_{S_2}. The following lemma states this property.

LEMMA 1. *Let S_1, S_2, and S_3 be subsets of U such that $U = S_1 S_2 S_3$. If in a constraint grah representing ρ, S_3 separates S_1 from S_2, then $\forall x \in \rho_{S_3}$,*

$$n_\rho(S_3 = x) = n_{S_3 S_1}(S_3 = x) \cdot n_{S_3 S_2}(S_3 = x). \quad \blacksquare$$

A similar product form holds for a tuple in ρ:

LEMMA 2. *If, in a graph representing* ρ, X_j *separates* $S_1 = X_1 \cdots X_{j-1}$ *from* $S_2 = X_{j+1} \cdots X_n$ (*see Fig.* 6), *then* $\forall x \in \rho$,

$$n(x_j) = n(x_1, ..., x_{j-1}, x_j) \cdot n(x_j, x_{j+1}, ..., x_n). \tag{13}$$

Proof. Let $S_i' = X_j S_i$, $i = 1, 2$. Since $\langle X_1, ..., X_{j-1} | X_j | X_{j+1}, ..., X_n \rangle$ then from Lemma 1,

$$\forall x_j \in \mathrm{Dom}(X_j), \qquad n(x_j) = n_{S_1'}(x_j) \cdot n_{S_2'}(x_j). \tag{14}$$

However,

a. $n_{S_1'}(x_j) = n(x_j, x_{j+1}, ..., x_n)$, and
b. $n_{S_2'}(x_j) = n(x_1, ..., x_j)$.

Equality (a) is true since $(x_j, x_{j+1}, ..., x_n) \in \rho_{S_2'}$ will appear in ρ as many times as $X_j = x_j$ appears in $\rho_{S_1'}(x_j)$. The same argument works for (b). Substituting (a) and (b) in (14) yields (13). ∎

An immediate extension of Lemma 2 is obtained if, instead of one separating variable, we have a chain or any subset of separating variables. In particular, we state the following corollary:

COROLLARY 1. *If in a graph representing* ρ, $X_j, ..., X_{j+i}$ *separate* $S_1 = X_1 \cdots X_{j+1}$ *from* $S_2 = X_{j+i+1} \cdots X_n$, *then* $\forall x \in \rho$,

$$n(x_{j+1}, ..., x_{j+i}) = n(x_1, ..., x_{j-1}, x_j, ..., x_{j+i}) \cdot n(x_j, x_{j+1}, ..., x_n). \tag{15}$$

The proof is based on Lemma 1 and follows the same steps as Lemma 2.

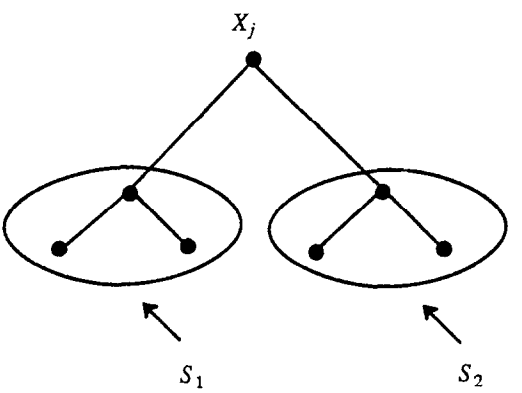


FIG. 6. $\langle S_1 | X_j | S_2 \rangle$.

LEMMA 3. *if S is a subtree of T rooted at X_i, and if X_i separates S_1 from S_2 (see Fig. 7), then $\forall \bar{x} \in S$,*

$$n(\bar{x}) = \frac{n(\bar{x} \mid S_1 X_i) \cdot n(\bar{x} \mid S_2 X_i)}{n(x_i)}, \tag{16}$$

where $\bar{x} \mid S$ denotes the projection of tuple $\bar{x}$ on the attributes in S.

Proof. From Lemma 2,

$$\forall \bar{x} \in S, \qquad 1 = \frac{n_S(\bar{x} \mid S_1 X_i) \cdot n_S(\bar{x} \mid S_2 X_i)}{n_S(x_i)}. \tag{17}$$

Let $S' = (U - S)\, X_i$. From the tree-structure we obtain

$$n(x_i) = n_{S'}(x_i) \cdot n_S(x_i). \tag{18}$$

Also, for both S_1 and S_2 it holds that

$$n(\bar{x} \mid S_i X_i) = n_S(\bar{x} \mid S_i X_i) \cdot n_{S'}(x_i). \tag{19}$$

Since also

$$n(\bar{x}) = n_{S'}(x_i), \tag{20}$$

we obtain

$$n(\bar{x}) = \frac{n(\bar{x} \mid S_1 X_i) \cdot n(\bar{x} \mid S_2 X_i)}{n(x_i)}. \quad \blacksquare \tag{21}$$

LEMMA 4. *Let ρ be a relation on $X_1 \cdots X_n$ represented by tree T and let S be a subtree of T rooted at X_i with $X_{j(i)}$ as its parent node (see Fig. 8), then $\forall \bar{x} = (x_{i_1}, \ldots, x_{i_l}) \in \rho_{S'}$,*

$$n(\bar{x}) = \frac{n(x_i, x_{j(i)}) \cdot n(\bar{x} \mid S)}{n(x_i)}, \tag{22}$$

where S' is the union of S with $X_{j(i)}$.

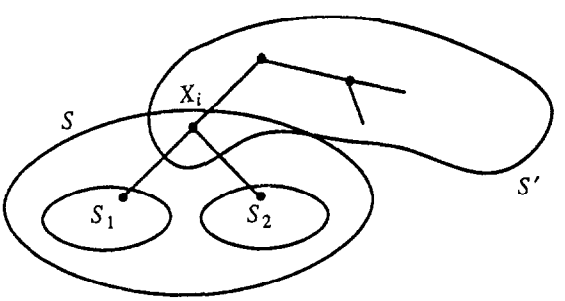


FIGURE 7

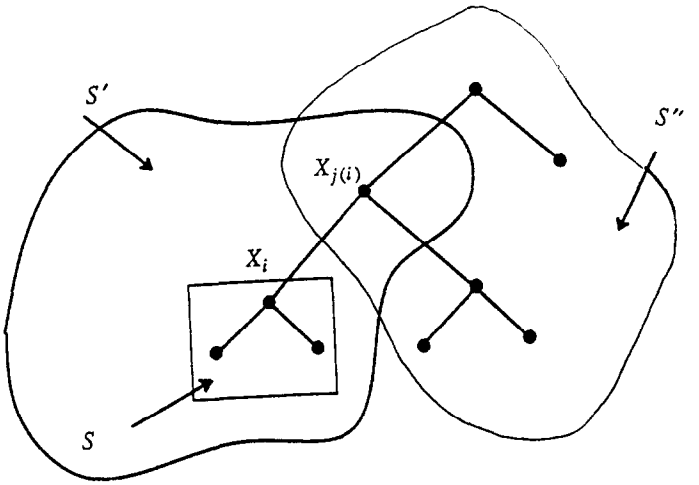


FIGURE 8

Proof. Let $S'' = U - S$, $\bar{x}$ be a member of $\rho_{S'}$, and let $\text{ext}(\bar{x})$ be an arbitrary extension of $\bar{x}$ to all attributes s.t. $\text{ext}(\bar{x}) \in \rho$. Since X_i separates S from S'', from Lemma 2 we obtain

$$n(x_i) = n(\bar{x} \mid S) \cdot n(\text{ext}(\bar{x}) \mid S'' X_i). \tag{23}$$

Also from Corollary 1 we obtain

$$n(x_i, x_{j(i)}) = n(\bar{x}) \cdot n(\text{ext}(\bar{x}) \mid S'' X_i). \tag{24}$$

Substituting (24) in (23) and some manipulation yields the desired result. ∎

Proof of Theorem 3. Theorem 3 can be proved by recursively applying Lemmas 2, 3, and 4. Given an n-tuple $x \in \rho$, then for x_1, the root value, we first apply Lemma 2 to yield

$$1 = n(x_{i_1}, ..., x_1, ..., x_{i_n}) = \frac{n(x_{i_1}, ..., x_1) \cdot n(x_1, ..., x_{i_n})}{n(x_1)}, \tag{25}$$

where the indices in the first and in the second tuples correspond to two subtrees separated by X_1. We continue to decompose each part of the numerator by applying Lemma 3 and Lemma 4 as necessary until we have only single and pairs of n-quantities. In this decomposition, each parent node appears $b-1$ times in the denominator when b is its degree in the tree. For all parents accept X_1, $b-1$ is also the number of children. We therefore obtain

$$1 = n(x_1) \prod_{i=1}^{n} \frac{n(x_i, x_{j(i)})}{n(x_{j(i)})}, \tag{26}$$

which completes the proof. ∎

We are now ready to state and prove the promised theorem 4.

THEOREM 4. *For any relation ρ, if $\rho = \rho^T$ then $U_\rho = (U_\rho)^T$.*

Proof. Let T be a tree such that $\rho = \rho^T$. By definition, U_ρ satisfies: $\forall x \in \rho$, $U_\rho(x) = 1/|\rho|$. We have to show, therefore, that $\forall x \in \rho$, $(U_\rho)^T(x) = 1/|\rho|$. By definition,

$$(U_\rho)^T(x) = \prod_{(X_i, X_{j(i)}) \in T} P(x_i | x_{j(i)}). \tag{27}$$

The conditional probabilities for U_ρ satisfies

$$P(x_i | x_j) = \frac{n(x_i, x_j)}{n(x_j)} \tag{28}$$

and

$$P(x_1) = \frac{n(x_1)}{|\rho|}. \tag{29}$$

Substituting (28) and (29) in (27) we obtain

$$(U_\rho)^T(x) = \frac{n(x_1)}{|\rho|} \prod_{(X_i, X_{j(i)}) \in T} \frac{n(x_i, x_{j(i)})}{n(x_{j(i)})}. \tag{30}$$

Finally, from Theorem 3 and (30) it follows that

$$(U_\rho)^T(x) = \frac{1}{|\rho|} = U_\rho(x), \quad \blacksquare \tag{31}$$

Although Theorems 3 and 4 could also be proved using maximum-entropy considerations [12, 14] the derivation we showed uses only structural properties of the relation and reveals interesting features, characteristic of tree-decomposable relations.

4. AN ALGORITHM FOR TREE DECOMPOSITION

Theorems 3 and 4 guarantee that a relation ρ has a tree-decomposition iff U_ρ is tree-dependent distribution. From Theorem 1 we know that a tree supporting a tree-dependent distribution is generated by MWST algorithm using the arc weights given in (3). If we substitute the conditional probabilities for the uniform distribution (given in (28) and (29)) in (3), we obtain the following arc-weights:

$$m(X_i, X_j) = \frac{1}{|\rho|} \sum_{(x_i, x_j) \in \rho_{x_i x_j}} n(x_i, x_j) \log \frac{n(x_i, x_j)}{n(x_i)\, n(x_j)}. \tag{32}$$

Since the quantity $|\rho|$ is common to all weights it can be ignored.

The tree-decomposition algorithm (described next), denoted also the MWST decomposition, takes a relation ρ and returns a set of tree-structured binary relations, which are guaranteed to be lossless if the relation is tree-decomposable.

TREE-GENERATION ALGORITHM. a. Compute the basic quantities: $n(x_i)$ and $n(x_i, x_j)$.

b. For every two attributes X_i, X_j compute the weights $m(X_i, X_j)$ given in (32).

c. Find a *maximum weight spanning tree* of the complete graph w.r.t. the above arc weights.

d. For each pair of attributes that corresponds to an arc in the selected tree find the associated relation by projecting the global relation on that pair.

The complexity of the algorithm is $O((l+\log n)\, n^2)$, where n is the number of attributes and l is the size of the relation. This can be shown by following its individual steps. The computation of part (a) can be completed in $O(ln^2)$ steps. Part (b) is bounded by $O(n^2 l)$ since the number of weights needed to be computed is $O(n^2)$, and each computation can take at most l steps. Since the MWST algorithm of part (c) takes just $O(n^2 \log n)$ steps [7], the total complexity is $O((l+\log n)\, n^2)$.

To verify that the generated tree represents the input relation, we can compute the number of n-tuples represented by the tree-decomposition and compare it to the size of the given relation. If the two numbers are equal, the database losslessly represents the relation. Otherwise, we know that no tree representation exists. The size of a relation represented by the tree can be computed in linear time [4], and (for the sake of completeness) an algorithm for doing so is henceforth described.

Given a directed tree (arcs are directed from a parent to its children), choose an ordering, d, on the attributes, such that a parent always precedes all its children. Let $N(x_{jt})$ stands for the size of the projection of ρ on the variables in the subtree rooted at X_j, after selecting those with $X_j = x_{jt}$. Consider a node X_j with all its child successor nodes as in Fig. 9.

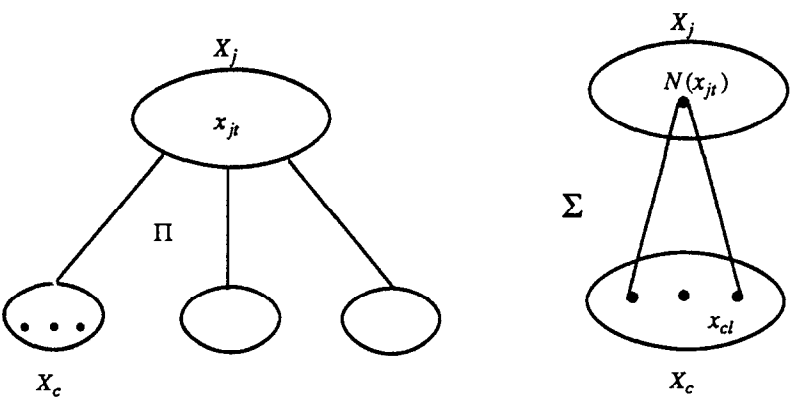


FIG. 9. Schematic computation of relation size on trees.

Looking first on the association between X_j and a specific child node X_c, it is clear that the value x_{jt} can participate in a consistent tuple with each compatible value of X_c, no matter what values are assigned to variables in the subtree rooted at X_c. Therefore, the number of tuples consistent with x_{jt} in the subtree rooted at X_c (i.e., in the projection of ρ on the variables in the subtree rooted at X_c), denoted $N_{X_c}(x_{jt})$, is a sum of those contributed by each compatible value of X_c. Namely,

$$N_{X_c}(x_{jt}) = \sum_{\{x_{cl} \in \mathrm{Dom}(X_c) \mid (x_{jt}, x_{cl}) \in \rho_{X_j X_c}\}} N(x_{cl}). \tag{33}$$

Since the partial tuples associated with different successor nodes can be joined (all of them have $X_j = x_{jt}$), the size of the joined relation, restricted to $X_j = x_{jt}$ and to the successors of X_j, will be the product of the corresponding sizes. Therefore, $N(\cdot)$ satisfies the following recurrence:

$$N(x_{jt}) = \prod_{\{c \mid X_c \text{ is a child of } X_j\}} \sum_{\{x_{cl} \in \mathrm{Dom}(X_c) \mid (x_{jt}, x_{cl}) \in \rho_{X_j X_c}\}} N(x_{cl}). \tag{34}$$

From this recurrence it is clear that the computation of $N(x_{jt})$ has the following steps. For each value x_{jt}, transfer to X_j, from each of its children, the sum total of the counts computed for the child's values that are consistent with x_{jt}. The overall value of $N(x_{jt})$ will be computed later by multiplying the summations obtained from each of the children. The computation starts at the leaves, initialized to $N = 1$, and progresses towards the root. Each variable performs the counting only after all its child nodes computed their counts using the procedure COUNT that follows.

The procedure COUNT peforms the calculation according to (34). The algorithm terminates when the root is assigned counts for all its values. The COUNT procedure is next defined for a parent node X_p and all its children $X_1, ..., X_r$,

COUNT($X_p, X_1, X_2, ..., X_r$)

1. begin
2. For each (X_p, X_l) do
3. For each $x_{p_s} \in \mathrm{Dom}(X_p)$ (for each value of X_p) do
4. $n_l(x_{p_s}) = \sum_{x_{l_t}:(x_{p_s}, x_{l_t}) \in \rho_{X_p X_l}} N(x_{l_t})$
5. end
6. For each $x_{p_s} \in \mathrm{Dom}(X_p)$ do
7. $N(x_{p_s}) = \prod_{l:X_l \text{ a child}} n_l(x_{p_s})$
8. end
9. end

Lines 3 and 4 take k^2 steps, where k bounds the domain size and, therefore, for each parent node, X_p, processing takes $k^2 \cdot \deg(X_p)$ steps, where $\deg(X)$ denotes the degree of X in the tree. Thus, the counting for all n variables in the subtree adds up to $O(nk^2)$.

5. Precision

The MWST algorithm, which is part of the tree-decomposing algorithm, bases its selection on the weights $m(X_i, X_j)$ given in (32). Therefore, we should compute the precision (i.e., the number of binary digits) required to reliably compare the weights. Since the weights are used merely to establish relative order, we can eliminate the logarithm function in (32) and use as weights, 2^m instead of m, thus maintaining the same relative order. The new weights $m' = 2^m$ are expressed by

$$m'(X_i, X_j) = \prod_{(x_i, x_j) \in \rho_{x_i x_j}} \left(\frac{n(x_i, x_j)}{n(x_i)\, n(x_j)} \right)^{n(x_i, x_j)}. \tag{35}$$

Since

$$\frac{n(x_i, x_j)}{n(x_i)\, n(x_j)} \leqslant 1, \tag{36}$$

we get that always $m' \leqslant 1$. In order to get a lower bound we use the fact that all the counts $n(x_i)$ and $n(x_i, x_j)$ are bounded by l, the size of the input relation, to obtain

$$m' \geqslant \left(\left(\frac{1}{l^2} \right)^l \right)^l = l^{-2l^2}. \tag{37}$$

Representing numbers of this size requires $2l^2 \log l$ binary digits. Hence, the precision needed for reliable decomposition requires temporary storage roughly quadratic in the size of the relation itself.

6. Tree- Decomposition as Approximation

It is clear that there is a very small chance that a random relation has a lossless decomposition into a tree of binary relations. If we take the set of binary relations generated by projecting the relation on *each pair of variables*, we get what Montanari called the *minimal network* [16] of the relation. He showed that the relation represented by the minimal network (i.e., the relation generated by joining all the binary relations in the minimal network) yields the best approximation to the original relation among all networks of binary relations. Clearly any tree decomposition is just an approximation to the minimal network's relation. Since the number of possible relations having n variables and k values each, is 2^k, while the number of possible networks of binary relations is roughly $2^{k^2 n^2}$ (the number of possible tree relations is about $O(n^{k^2 n})$), there is a very small probability that an arbitrary relation will have a lossless decomposition by the minimal network, and more so by any tree decomposition.

When no tree decomposition exists, the decomposition algorithm still produces a tree decomposition which is not lossless but can be regarded as an approximation to ρ. In the context of probability distribution we know that the tree, T, generated by the algorithm determines a tree-dependent distribution $(U_\rho)^T$ which is the best approximation to U_ρ with respect to proximity measure (2). However, it is not clear how this proximity measure can be translated into a meaningful distance measure for relations. The distance measure (2) for U_ρ is given by

$$D(U_\rho, (U_\rho)^T) = \frac{1}{|\rho|} \sum_{x \in \rho} \log \frac{1}{|\rho|(U_\rho)^T(x)}, \tag{38}$$

and after some manipulation one can show that minimizing (38) is equivalent to maximizing the product

$$\prod_{x \in \rho} (U_\rho)^T(x). \tag{39}$$

Substituting (30) in (39), results in

$$D'(T) = \prod_{x \in \rho} \frac{1}{(|\rho|)} n(x_1) \prod_{(x_i, x_{j(i)}) \in T} \frac{n(x_i, x_{j(i)})}{n(x_{j(i)})}. \tag{40}$$

Therefore, the algorithm finds a tree, T, which maximizes $D'(T)$.

We cannot, however (as of this writing), make any theoretical claim regarding the relationship between this measure and a more intuitive measure such as the number of tuples in the approximating relation. Setting aside the issue of how well a relation can be approximated by a tree, we still would like to verify that the approximation provided by the MWST method is better than random tree decompositions. To gain insight we performed an initial set of experiments the results of which are summarized in Figs. 10–12. In the table of Fig. 10, each row shows the results associated with one randomly generated relation. We experimented with relations having 4–15 variables each having two values. For each relation we show its size (r-size), the size of the relation generated from the MWST decomposition (*mwst*-size) and the size of relations generated from random tree decompositions ($\#Ti$). To our satisfaction, in all cases tried, the MWST decomposition was better or equal to a random tree decomposition. For larger relations the binary projections permitted all four pairs and, therefore, the approximation is equally bad for all trees. Later, when we experimented with multi-valued relations (not reported here), we could scarcely see cases where the MWST decomposition was outperformed by a randomly generated tree.

Intuitively, one may think that an approximating tree decomposition should consist of the tightest binary relations. We followed this intuition and in Fig. 11 we compare the MWST decomposition and random trees decompositions with the tree generated by the minimum weight spanning tree algorithm, where the *weight of each arc is the size of the projected binary relation* (called minimum projection size

#-vars	r-size	mwst-size	T1	T2	T3	T4	T5
4	2	2	4	4	4	2	4
4	3	3	4	4	4	3	4
6	7	12	24	16	14	15	14
6	5	14	18	24	24	20	24
6	7	16	24	24	20	20	20
7	13	48	64	64	64	64	48
8	25	128	128	128	128	128	128
8	8	20	40	48	48	40	40
8	42	256	256	256	256	256	256
8	12	96	96	96	128	96	128
8	15	128	128	128	128	128	128
8	8	20	36	48	36	48	64
10	6	31	96	78	144	46	104
10	6	60	116	120	126	216	288
12	21	2560	2560	3072	3072	2560	3072
12	20	4096	4096	4096	4096	4096	4096
12	10	352	1408	1008	2304	900	1296
12	12	960	3072	4096	4096	2304	3072
15	2	2	16	64	8	64	32
15	4	7	30	24	32	32	48
15	6	16	60	72	72	72	64
15	5	7	72	128	16	144	144
15	5	15	240	72	160	200	144
15	2	2	16	32	16	16	32
15	4	10	192	160	169	270	30
15	3	3	72	96	40	12	96
15	5	52	888	468	756	480	864

FIGURE 10

#-vars	r-size	mwst-size	mpst	T1	T2	T3	T4	T5
9	4	5	128	48	26	32	14	16
11	9	96	1024	400	270	432	240	432
11	14	336	1024	400	640	640	576	576
9	6	34	256	160	96	78	96	90
9	6	18	256	96	144	120	72	112
8	2	2	8	8	8	4	4	8

FIG. 11. Comparing mwst decomposition with minimum projection size tree (mpst).

tree–mpst). To our surprise, the trees generated that way provided the worse approximations even compared to random tree decompositions.

As noted earlier, it is quite unfair to judge the approximating power of the tree decomposition on an arbitrary relation since even the minimal network which is the best possible approximation will do poorly. We therefore evaluated the MWST decomposition for relations which are representable by networks of binary relations (in this case a random network of binary relations was generated and the relation associated with them was computed and served as input to the decomposition algorithm). The results, tabulated in Fig. 12, show that the MWST decomposition is lossless in most cases while in others it provides a very good approximation to the relation, much better than any random tree decomposition.

The quality of the approximation can be enhanced by using also a tree approximation of the *Complement* of ρ, $\bar{\rho} = \{x \mid x \notin \rho\}$, which yields a lower bound to ρ. The weights of the arcs associated with the complement relation as well as their corresponding binary relations do not require that the complement relation be explicitly generated. Assume that, a priori, each variable has k possible values, and let $\bar{n}$ denote the n-qauntities of the complement relation. The equality

$$\bar{n}(x_i, x_j) = k^{n-2} - n(x_i, x_j) \tag{41}$$

holds, since any $(n-2)$-tuple of all the attributes excluding X_i and X_j, which is not consistent with (x_i, x_j) will be consistent with (x_i, x_j) in $\bar{\rho}$. Similarly,

$$\bar{n}(x_i) = k^{n-1} - n(x_i). \tag{42}$$

Thus, the weights $\bar{m}$ are given by

$$\bar{m}(X_i, X_j) = \sum_{(x_i, x_j) \in \mathrm{Dom}(X_i) \times \mathrm{Dom}(X_j)} (k^{n-2} - n(x_i, x_j)) \times \log \frac{k^{n-2} - n(x_i, x_j)}{(k^{n-1} - n(x_i))(k^{n-1} - n(x_j))}. \tag{43}$$

#-vars	r-size	mwst-size	mpst	T1	T2	T3	T4	T5
10	52	66	206	512	360	324	384	360
11	64	64	128	128	128	128	96	96
9	8	8	16	12	10	12	12	16
9	16	16	32	20	24	24	16	24
8	15	17	128	54	56	64	42	72
7	48	48	60	128	72	96	96	72
7	36	36	68	80	96	64	96	64
7	6	6	16	12	8	8	8	9
6	28	28	28	36	48	40	48	40
5	16	16	20	24	24	32	24	16
5	14	14	16	18	20	20	24	20
4	9	9	9	9	12	16	16	9
3	5	5	5	6	6	5	6	6

FIG. 12. Comparing mwst-decomposition for relations represenatable by a binary network of relations.

With these weights the minimum spanning tree algorithm determines a tree $\bar{T}$, so that $\bar{\rho}^{\bar{T}}$ can be generated. The binary relations associated with each arc of this tree contain the Cartesian product of the domains of the two attributes *excluding* those pairs (x_i, x_j) for which $\bar{n}(x_i, x_j)$ is 0. Namely, for each $(X_i, X_j) \in \bar{T}$,

$$\bar{\rho}_{X_iX_j} = \{(x_i, x_j) | (x_i, x_j) \in \mathrm{Dom}(X_i) \times \mathrm{Dom}(X_j); \bar{n}(x_i, x_j) > 0\}. \tag{44}$$

ρ^T and $\bar{\rho}^{\bar{T}}$ provide two bounding sets for the relation ρ that have a compact representation, and the membership of a tuple in each set can be efficiently determined. A tuple which is not in ρ^T is definitely not in ρ and a tuple which is not in $\bar{\rho}^{\bar{T}}$ is definitely in ρ (see Fig. 13). A tuple belonging to both sets is undetermined.

Notice, however, that the usefulness of the approximation by the complement relation is limited to large relations only. It is apparent (see (41) and (44)) that if the relation size, l, is smaller than k^{n-2} then the projection of the complement relation on each pair of variables allows all possible combinations of pairs and the approximation generated by such binary relations is the universal relation, thus providing no information. By the same token, if the size of the relation itself is very big, i.e., if $l \geqslant (k^2 - 1)\, k^{n-2}$ then approximating the relation itself by binary relations will yield the worse approximation—the universal relation. In this case the approximation by use of the complement relation may be more informative. In summary, if we divide the possible relation sizes into three regions: 1. $l \leqslant k^{n-2}$; 2. $k^{n-2} < l \leqslant (k^2 - 1)\, k^{n-2}$; 3. $l > (k^2 - 1)\, k^{n-2}$, then only the relation should be approximated in the first region, both the relation and its complement should be approximated in the second region, and only the complement relation should be approximated in the third region.

An important feature of these approximations is that their quality can be determined in advance. The quality of the tree approximation ρ^T can be measured by the ratio of its relation size to l.

The computation of $\bar{\rho}^{\bar{T}}$ does not involve a precision problem, since it should only be used when $l \geqslant k^{n-2}$ and in these cases the size of the complement relation is no larger than $k^2 l$, where k is the number of values. The precision required for computing the arc weights of the complement relation are therefore roughly quadratic in that number.

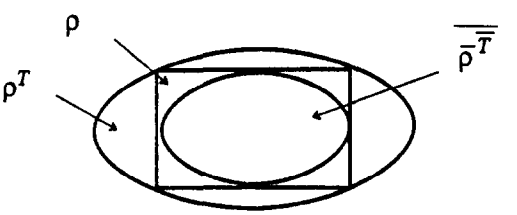


FIGURE 13

7. Extension to Acyclic Schemes

The method described above is restricted to representing, or approximating, a relation by a tree of *binary* relations. It can be generalized to allow relations of higher arity in the decomposition as long as they are interconnected via a tree structure. Such decompositions are called *join-trees* [11].

A set of relations $R_1, ..., R_t$, can be associated with a graph, called a *clique graph*, in which the attributes are represented by nodes and pairs of attributes which belong to the same relation are connected by an arc. Thus, every relation is associated with a clique. A database scheme whose clique graph is chordal (i.e., every cycle of length at least four has a chord), and whose maximal cliques correspond to relations in the scheme, is called an *acyclic scheme* [2]. There exists an underlying join-tree for such a scheme, namely, the maximal cliques can be connected by a tree structure, whose topology dictates an efficient procedure for a lossless recovery of the whole relation. The existence of a join-tree enables an efficient query processing algorithms and therefore acyclic databases are desirable decomposition schemes [2].

Since a chordal graph can be associated with an acyclic database scheme whose relations are determined by the maximal cliques (i.e., those which are not contained in another clique) of the graph, it seems reasonable to extend the decomposition target from simple trees, representing only binary relations, to join-trees, having relations (cliques) of varying sizes. Since cliques of smaller sizes are preferable, we will characterize the join-trees by the cardinality of the cliques and will consider a special class of chordal graphs called *k-trees*. A *k-tree* is a chordal graph whose maximal cliques are of size $k+1$ and it can be defined constructively as follows:

1. A complete graph with k vertices is a k-tree.
2. A k-tree with r vertices can be extended to $r+1$ vertices, by connecting the new vertex to all the vertices in a clique of size k.

In particular, 1-trees are ordinary trees. The addition of each vertex (step 2) generates a new clique of size $k+1$ and, by associating each new clique with one "parent clique" which shares k vertices with it, we get a join-tree. Figure 14a presents a 2-tree which could be constructed in the order A, B, C, D, E, F.

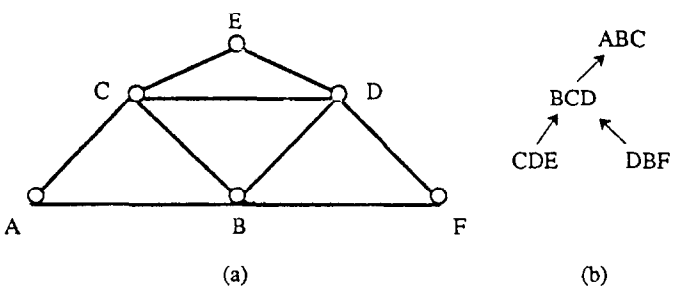


FIGURE 14

A possible join tree among the cliques *ABC*, *BCD*, *DFB*, and *CDE* is given in Fig. 14b. For a detailed discussion of the properties of k-trees see [1].

Since k-trees represent acyclic schemes, we consider them as a desirable target for decomposition. Equation (12) can be extended to any join-tree of an acyclic scheme [13]. Given a directed join-tree with relations $R_1, ..., R_t$, when R_1 is the root of the tree, the relation which can be recovered from this scheme satisfies

$$\forall \bar{x} \in \rho,\ \bar{x} = x_1, ..., x_n, \qquad 1 = n(\bar{x} \mid R_1) \prod_{i=2}^{t} \frac{n(\bar{x} \mid R_i)}{n(\bar{x} \mid R_i \cap R_{j(i)})}, \tag{45}$$

where R_i represents the set of attributes in the ith relation, $(\bar{x} \mid R_i)$ denotes the projection of n-tuple $\bar{x}$ on the attributes in R_i, and $R_{j(i)}$ is the parent of R_i in the join-tree. We can show that if with each join-tree, T, we associate a mapping

$$T(\bar{x}) = n(\bar{x} \mid R_1) \prod_{(R_i,\ R_{j(i)}) \in T} \frac{n(\bar{x} \mid R_i)}{n(\bar{x} \mid R_i \cap R_{j(i)})}, \tag{46}$$

and a measure

$$F_\rho(T) = \sum_{\bar{x} \in \rho} \log T(\bar{x}), \tag{47}$$

then $F_\rho(T)$ satisfies

$$F_\rho(T) = m(R_1) + \sum_{(R_i,\ R_{j(i)}) \in T} (m(R_i) - m(R_i \cap R_{j(i)})), \tag{48}$$

where

$$m(R_i) = \sum_{x \in \rho_{R_i}} n(x) \log n(x). \tag{49}$$

Associating with each relation ρ_{R_i} the weight

$$w(R_i) = m(R_i) - m(R_i \cap R_{j(i)}) \tag{50}$$

(for the root, R_1, the parent relation is empty), we obtain

$$F_\rho(T) = \sum_{i=1}^{t} w(R_i). \tag{51}$$

In particular, every join-tree of a k-tree has an F value which can be computed by (50). We can prove that:

THEOREM 5. *A relation ρ has an acyclic decomposition into a k-tree, having a join-tree, T, iff T maximizes F over all other k-trees (and their join-trees).*

The theorem can be proved in the same way as in the binary tree-decomposition case, however, here we will present proof of one part of the theorem without using the analogy between probabilities and relations. We will show that if T is a join-tree representing ρ, then T maximizes F.

Proof of Theorem 5. Since F is a concave and symmetric function on $V = \{(T(\bar{x}_1), ..., T(\bar{x}_l)) \mid T \in \textit{JOIN-TREES OF K-TREES}\}$ and it is bounded by a symmetric constraint $\sum_{\bar{x}} T(\bar{x}) \leqslant l$ [3], where l is the size of the relation ρ, F's extremum (maximum) is achieved when $T(\bar{x})$ are all equal. Moreover, since F is monotone w.r.t. each of its components, if there exist T such that $\forall \bar{x}$, $T(\bar{x}) = 1$, F will attain its maximum in this T. ∎

We, therefore, seek a k-tree whose directed join-tree has a maximal sum of weights. Unfortunately, k-trees do not possess some of the nice properties of ordinary trees. In particular, there is no greedy algorithm that enables the determination of a k-tree of maximum sum of weights (i.e., k-trees are not matroids [9]). Determining the maximum weight k-tree may require exhaustive search among all possible k-trees. Alternatively, one can always use a greedy algorithm heuristically and arrive at a k-tree which has a (hopefully) good but non-optimal F value.

Such an algorithm can generate the k-tree incrementally. At each step one vertex is selected and connected to a clique of size k already in the tree. This determines a new clique whose parent will be one of the old cliques that shares k vertices with it. The vertex chosen is one which contributes maximum weight according to (50). When $k = 1$ the algorithm is the same as the binary-tree decomposition described before, and only in this case is the decomposition guaranteed to be lossless, if one exists. Otherwise, the algorithm is heuristic and a lossless decomposition into a k-tree may exist even if the algorithm does not arrive at such a decomposition.

8. Conclusion

We have presented an efficient algorithm for decomposing an n-ary relation into a tree of binary relations, and a simple test for checking whether or not the tree formed represents the relation. If such a tree decomposition exists, the algorithm is guaranteed to find one. Otherwise, the tree generated will fail the test, indicating that no tree decomposition into binary relations exists. We then discuss the use of tree decomposition of the relation and its complement as an approximation to the relation. Finally, we proposed a heuristic algorithm for decomposing any relation into an acyclic decomposition of bounded arity by extending the binary-tree decomposition scheme into a k-tree decomposition.

The work reported here has its motivation in the area of constraint satisfaction problems (CSPs), which have many applications in AI [10]. Constraint satisfaction involves the assignment of values to variables subject to a set of constraints, where each constraint is an i-ary relation on a subset of i variables ($i \leqslant n$), and the task is to find one or all solutions. Thus, a network of constraints is an instance of a data base scheme, variables in a CSP are attributes in database, and the task of finding all solutions is equivalent to creating the join of all relation instances in the scheme. As in databases, it was realized in CSPs that tree-like structures provide a useful representation that is accompanied by efficient processing algorithms.

Normally in a CSP environment the relation itself is not available for decomposition (this is the target of processing) but in applications involving large networks representing knowledge that is going to be queried repeatedly, it may be worthwhile to manipulate the structure of the representation, and the tree-decomposition algorithm provides one such tool. Recently, we had developed a greedy "qualitative" tree-decomposition scheme which can be applied directly to the constraint network (in "Proceedings of AAAI-90, Boston, MA").

ACKNOWLEDGMENTS

I thank Avi Dechter and Judea Pearl for providing many helpful comments and for reading this manuscript. I thank the students, Irit Moshe and Ronit Nadam, who implemented the tree-decomposition algorithm and provided the experimental results presented in the paper. I also thank the referees of this paper.

REFERENCES

1. S. ARNBORG, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey, *BIT* **25** (1985), 2–23.
2. C. BEERI, R. FAGIN, D. MAIER, AND M. YANNAKAKIS, On the desirability of acyclic database schemes, *J. Assoc. Comput. Mach.* **30**, No. 3 (1983), 479–513.
3. C. K. CHOW AND C. N. LIU, Approximating discrete probability distributions with dependence trees, *IEEE Trans. Inform. Theory* (1968), 462–467.
4. R. DECHTER AND J. PEARL, Network-based heuristics for constraint-satisfaction problems, *Artif. Intell. J.* **34**, No. 1 (1987), 1–38.
5. S. EVEN, "Graph Algorithms," Comput. Sci. Press, Rockville, MD, 1979.
6. R. FAGIN, Multivalued dependencies and a new form for relational databases, *ACM Trans. Database Systems* **2**, No. 3 (1977), 262–278.
7. J. B. KRUSKAL, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Amer. Math. Soc.* **7** (1956), 48–50.
8. S. KULLBACK AND R. A. LEIBLER, Information and sufficiency, *Ann. Math. Statist.* **22** (1951), 79–86.
9. E. L. LAWLER, "Combinatorial Optimization, Networks and Matroids," Holt, Rinehart, & Winston, New York, 1976.
10. A. K. MACKWORTH, Consistency in networks of relations, *Artif. Intell.* **8**, No. 1 (1977), 99–118.
11. D. MAIER, "The Theory of Relational Databases," Comput. Sci. Press, Rockville, MD, 1983.
12. F. M. MALVESTUTO, Decomposing complex contingency tables to reduce storage requirements, *in* "Proceedings, International Workshop on Statistical & Scientific Database Management, Luxenburg, 1986."
13. F. M. MALVESTUTO Modelling large bases of categorical data with acyclic Schemes, Rome, 1986.
14. F. M. MALVESTUTO, Statistical treatment of the information content of a database, *Inform. Systems* **11**, No. 3 (1986), 211–223.
15. F. M. MALVESTUTO, Answering queries in categorical databases, *in* "Proceedings, Sixth Conference on the Principles of Database Systems, San Diego, CA, 1987," pp. 87–96.
16. U. MONTANARI, Networks of constraints: Fundamental properties and applications to picture processing, *Inform. Sci.* **7** (1974), 95–132.
17. J. PEARL AND A. PAZ, On the logic of representing dependencies by graphs, *in* "Proceedings, AI-86, Canadian Conference, Montreal, Canada, 1986."
18. J. PEARL, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," Morgan and Kauffman, Palo Alto, CA, 1988.
19. J. E. SHORE, Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy, *IEEE Trans. Inform. Theory* **IT-26** (1980), 26–37.