Maximal Prime Subgraph Decomposition of Bayesian Networks: A Relational Database Perspective

Dan Wu
School of Computer Science
University of Windsor
Windsor Ontario
Canada N9B 3P4

Michael Wong
Department of Computer Science
University of Regina
Regina Saskatchewan
Canada S4S 0A2

Abstract

A maximal prime subgraph decomposition junction tree (MPD-JT) is a useful computational structure that facilitates lazy propagation in Bayesian networks (BNs). A graphical method was proposed to construct an MPD-JT from a BN. In this paper, we present a new method from a relational database (RDB) perspective which sheds light on the semantic meaning of the previously proposed graphical algorithm.

1. Introduction

Bayesian networks have been widely used for reasoning with uncertainty (Pearl 1988). Normally, a Bayesian network is transformed into a junction tree on which probabilistic reasoning is conducted (Lauritzen & Spiegelhalter 1988; Huang & Darwiche 1996). Various attempts have been made to improve the efficiency of probabilistic reasoning, for instance, the *lazy propagation* method (Madsen & Jensen 1998).

Very recently, it was proposed to transform a Bayesian network *not* into a junction tree, but into a *maximal prime subgraph decomposition junction tree* (MPD-JT) on which the lazy propagation can be greatly facilitated. A graphical algorithm has been developed to construct the MPD-JT (Olesen & Madsen 2002).

On the other hand, it has long been noted that there exists an intriguing relationship between BNs and RDBs (Wen 1991) such that many problems in BNs can be considered as similar problems in RDBs (Wong, Wu, & Butz 2002; Wong & Butz 2001). In this paper, we suggest a new algorithm for constructing an MPD-JT from a RDB perspective after carefully examining the graphical algorithm in (Olesen & Madsen 2002). By investigating the conditional independencies (CIs) encoded in an MPD-JT, we show that constructing an MPD-JT from a BN is equivalent to obtaining a conflict free set of CIs encoded in the BN. This new perspective makes it possible to apply a well-developed algorithm for constructing an acyclic database scheme in RDBs to constructing an MPD-JT in BNs. This new method sheds light on the semantic meaning of the graphical method in (Olesen & Madsen 2002) and further confirms the strong relationship between BNs and RDBs.

The paper is organized as follows. In Section 2, we review pertinent background. In Section 3, we discuss the relationship between BNs and RDBs which serves as the basis for the new proposed algorithm. In Section 4, we present the proposed new method and discuss its complexity. We conclude the paper in Section 5.

2. Background

Bayesian Networks

A *Bayesian network* (BN) defined over a set $V = \{A_i |\ i = 1, \ldots, n\}$ of finite discrete variables is a *directed acyclic graph* (DAG) denoted $\mathcal{D}$, augmented with a set of *conditional probability distributions* (CPDs). Each vertex in $\mathcal{D}$ corresponds one-to-one to a variable in V (we thus use the terms *vertex*, *node*, and *variable* interchangeably). The parents of a node A_i in $\mathcal{D}$ is denoted π_{A_i}. Each variable $A_i \in V$ is associated with a CPD $p(A_i|\pi_{A_i})$ such that the *joint probability distribution* (JPD) $p(V) = \prod_{A_i \in V} p(A_i|\pi_{A_i})$.

The DAG of a BN encodes CIs satisfied by the JPD $p(V)$. We use the notation $Y \Rightarrow\Rightarrow X|Z$, where X, Y, and Z are disjoint subsets of V, to denote that given Y, X and Z are conditionally independent (Pearl 1988). A CI is *full* if $XYZ = V$ (by XYZ, we mean $X \cup Y \cup Z$), otherwise, it is *embedded*.

Junction Trees

The DAG of a BN is normally transformed into a junction tree for probabilistic reasoning. The transformation consists of two graphical operations, namely, *moralization* and *triangulation* (Lauritzen & Spiegelhalter 1988). The moralized graph of a DAG $\mathcal{D}$, denoted $M^{\mathcal{D}}$, is an undirected graph obtained by connecting every pair of nodes with a common child which are not already connected in $\mathcal{D}$ and then dropping the directionality of all directed edges. $M^{\mathcal{D}}$ is then *triangulated* by adding a chord, called *fill-in* edge, to every cycle whose length is greater than three. A triangulation is *minimal* if removal of any fill-in edge will result in an untriangulated graph. The resulting triangulated graph is denoted $T^{\mathcal{D}}$. A *junction tree*, written as $\mathcal{T}$, is constructed by identifying all the cliques (i.e., maximal complete subgraphs) of $T^{\mathcal{D}}$ and arranging them as nodes of a tree to sat-

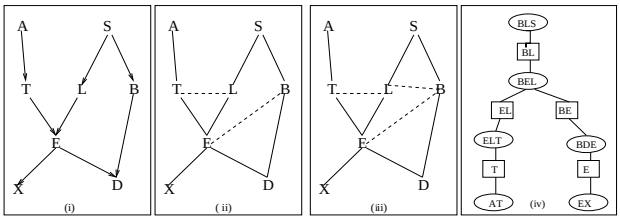

Figure 1: (i) The DAG $\mathcal{D}$ of the "Asia" BN . (ii) The moralized graph $M^{\mathcal{D}}$ of the DAG $\mathcal{D}$. (iii) The triangulated graph $T^{\mathcal{D}}$ of the DAG $\mathcal{D}$. (iv) The junction tree $\mathcal{T}$ constructed from $T^{\mathcal{D}}$.

isfy the *running intersection property* (Lauritzen & Spiegelhalter 1988), which requires that every clique C on the path between two cliques $C^{'}$ and $C^{''}$ in $\mathcal{T}$ contains $C^{'} \cap C^{''}$. Note that multiple junction trees may be produced from $T^{\mathcal{D}}$ on the arrangements of the cliques, satisfying the running intersection property. If cliques $C^{'}$ and $C^{''}$ are connected by an edge in $\mathcal{T}$, this edge is labelled with $C^{'} \cap C^{''}$. The set $\mathcal{S}^{\mathcal{T}} = \{C \cap C^{'} \mid \text{there is an edge between } C \text{ and } C^{'}\}$ is called the *separator set* of $\mathcal{T}$, each element in $\mathcal{S}$ is called a *separator* of $\mathcal{T}$.

Example 1 Consider the "Asia" BN (Lauritzen & Spiegelhalter 1988) whose DAG $\mathcal{D}$ is in Fig. 1 (i). Its moralized graph $M^{\mathcal{D}}$ and triangulated graph $T^{\mathcal{D}}$ are shown in Fig. 1 (ii) and (iii), respectively. The dotted edges $(T,\ L)$ [1] and $(E,\ B)$ in $M^{\mathcal{D}}$ in Fig. 1 (ii) were added during moralization. The fill-in edge $(L,\ B)$, dotted in Fig. 1 (iii), was added during triangulation. There are 6 cliques in the triangulated graph $T^{\mathcal{D}}$ and they have been arranged as a junction tree $\mathcal{T}$ in Fig. 1 (iv). Note that in Fig. 1 (iv), the oval represents a clique identified from $T^{\mathcal{D}}$, the square box represents a separator in $\mathcal{S}^{\mathcal{T}}$.

MPD Junction Trees

Very recently, it was suggested to transform a BN *not* into a junction tree, but into an MPD-JT to facilitate lazy propagation (Olesen & Madsen 2002). Before we introduce MPD-JT, the following pertinent graph terminologies are needed.

We use $G(V)$ (or G) to denote an undirected graph consisting of a finite set V of vertices.

Definition 1 Let the triplet $(V_1,\ S,\ V_2)$ denote a partition of V where $V_1 S V_2 = V$. If every path in $G(V)$ between $A_i \in V_1$ and $A_j \in V_2$ contains a vertex in S, then S is a *separator* of $G(V)$ with respect to $(V_1,\ S,\ V_2)$. Furthermore, if the subgraph induced by the separator S, i.e., $G(S)$, is complete, then S is a *complete separator* of $G(V)$ with respect to $(V_1,\ S,\ V_2)$.

Definition 2 A partition $(V_1,\ S,\ V_2)$ is a *decomposition* of a graph $G(V)$ if S is a complete separator. The complete separator S is further called a *minimal complete separator* if for any $S^{'} \subset S$, $(V_1,\ S^{'},\ V_2 \cup (S - S^{'}))$ is *not* a decomposition anymore.

[1] We use $(A_i,\ A_j)$ to denote a directed edge in a DAG.

Note that each separator in $\mathcal{S}^{\mathcal{T}}$ (i.e., the separator set of junction tree $\mathcal{T}$) is a minimal complete separator of the triangulated graph from which the junction tree $\mathcal{T}$ is constructed.

Definition 3 An undirected graph is a *prime* graph if it has no complete separator. If $G^{'}$ is a subgraph of G and $G^{'}$ is also a prime graph, then $G^{'}$ is a *prime subgraph* of G. $G(V^{'})$ is a *maximal prime subgraph* (mp-subgraph) of $G(V)$ if $G(V^{'})$ is a prime subgraph of $G(V)$ and for any other subgraph $G(V^{''})$ of $G(V)$, where $V^{'} \subset V^{''}$, $G(V^{''})$ is *not* a prime subgraph.

Definition 4 (Olesen & Madsen 2002) Let G be an undirected graph, its *mp- subgraph decomposition* (MPD) is the set of induced mp-subgraphs of G. A *MPD junction tree* (MPD-JT) of G is a junction tree whose nodes are the mp-subgraphs of G.

Definition 5 Consider a BN with its DAG $\mathcal{D}$, its MPD is the set of induced mp-subgraphs of the moralized graph $M^{\mathcal{D}}$ (Olesen & Madsen 2002). A *MPD-JT* of the BN is a junction tree whose nodes are the mp-subgraphs of $M^{\mathcal{D}}$.

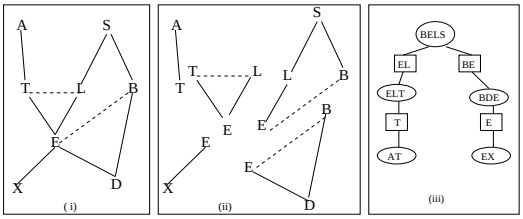

Figure 2: (i) The moralized graph $M^{\mathcal{D}}$ of the "Asia" BN in Fig. 1. (ii) The induced mp-subgraphs of $M^{\mathcal{D}}$. (iii) The MPD-JT of the "Asia" BN.

Example 2 Consider the moralized graph $M^{\mathcal{D}}$ of the "Asia" BN in Fig. 2 (i). $M^{\mathcal{D}}$ induces 5 mp-subgraphs as shown in Fig. 2 (ii). An MPD-JT of the "Asia" BN can be constructed by arranging these 5 mp-subgraphs to satisfy the running intersection property as shown in Fig. 2 (iii).

Junction Trees and Hypertrees

The notion of junction tree has a close tie with the notion of hypertree which is widely used in RDBs (Maier 1983).

Definition 6 A *hypergraph* is a pair $(\mathcal{N}, \mathcal{H})$, where $\mathcal{N}$ is a finite set of vertices and $\mathcal{H}$ is a set of *hyperedges* which are arbitrary subsets of $\mathcal{N}$ (Shafer 1991), that is, $\mathcal{H} = \{h_1,\ \ldots,\ h_n \mid h_i \subseteq \mathcal{N}\}$. We usually use $\mathcal{H}$ to denote the hypergraph $(\mathcal{N}, \mathcal{H})$.

Definition 7 We say that a hyperedge h_i in a hypergraph $\mathcal{H}$ is a *twig* if there exists another hyperedge h_j in $\mathcal{H}$, distinct from h_i, such that $(\bigcup_{h \in \mathcal{H} - \{h_i\}} h) \cap h_i = h_i \cap h_j$. We call any such h_j a *branch* for the twig h_i.

The notions of twig and branch capture the scenario that the intersection of a hyperedge, namely, the twig, with the rest of the hypergraph is contained by a single hyperedge, namely, the branch.

Definition 8 A hypergraph $\mathcal{H}$ is a *hypertree* (Shafer 1991) or *acyclic* if its hyperedges can be ordered, say $h_1, h_2, ..., h_n$, such that h_i is a twig in $\{h_1, h_2, ..., h_i\}$, for $i = 2, ..., n$. We call any such ordering a *tree (hypertree) construction ordering* for $\mathcal{H}$.

Definition 9 Given a tree construction ordering $h_1, h_2, ..., h_n$, we can choose, for i from 2 to n, an integer $j(i)$ such that $1 \leq j(i) \leq i-1$ and $h_{j(i)}$ is a branch for h_i in $\{h_1, h_2, ..., h_i\}$. We call a function $j(i)$ that satisfies this condition a *branching function* for $\mathcal{H}$ and the tree construction ordering $h_1,\ h_2,\ ...,\ h_n$.

Note that for a given hypertree, there might exist multiple tree construction orderings; for a given tree construction ordering, there might exist multiple choices of branching functions (Shafer 1991).

Given a tree construction ordering $h_1, h_2, ..., h_n$ for a hypertree $\mathcal{H}$ and a branching function $j(i)$ for this ordering, we can construct a set denoted $\mathcal{S}^{\mathcal{H}}$ whose elements are the intersections of the twigs and their respective branches, i.e., $\mathcal{S}^{\mathcal{H}} = \{h_{j(2)} \cap h_2, h_{j(3)} \cap h_3, ..., h_{j(n)} \cap h_n\}$. Note that $\mathcal{S}^{\mathcal{H}}$ is the same for any tree construction ordering of a given hypertree (Shafer 1991) and we call $\mathcal{S}^{\mathcal{H}}$ the *separator set* of $\mathcal{H}$. Note also that for any hypertree $\mathcal{H}$, there is a unique corresponding triangulated undirected graph denoted $G^{\mathcal{H}}$ which has the same nodes as $\mathcal{H}$ and whose edges are constructed by connecting every two nodes that belongs to the same hyperedge of $\mathcal{H}$. On the other hand, for any triangulated graph G, its cliques, being considered as hyperedges, constitute a unique hypertree denoted $\mathcal{H}_G$ (Maier 1983). In other words, there is a one-to-one correspondence between triangulated undirected graphs and hypertrees.

Since a hypertree corresponds to a triangulated undirected graph and a triangulated undirected graph may produce multiple junction trees, it is not surprising that given a hypertree, there exists a set of junction trees each of which corresponds to a particular tree construction ordering and a branching function (Shafer 1991). This property implies that hypertrees are more versatile than junction trees. It is also trivial to see that given a junction tree $\mathcal{T}$, there always exists a unique corresponding hypertree representation denoted $\mathcal{H}_{\mathcal{T}}$ whose hyperedges are the cliques in the junction tree. Therefore, we will treat a junction tree $\mathcal{T}$ as if it is a hypertree $\mathcal{H}_{\mathcal{T}}$ whenever appropriate. Furthermore, the separator set $\mathcal{S}^{\mathcal{H}}$ of the hypertree $\mathcal{H}_{\mathcal{T}}$ is exactly the same as the separator set $\mathcal{S}^{\mathcal{T}}$ of the junction tree $\mathcal{T}$, and both separator sets correspond exactly to the minimal complete separators of the triangulated graph $G^{\mathcal{H}_{\mathcal{T}}}$. Therefore, a junction tree in essence is a hypertree (with a particular tree constructing ordering and branching function) and the problem of constructing junction trees can be considered as a more general problem of constructing hypertrees.

Example 3 Consider the hypertree $\mathcal{H}$ in Figure 3 (i). It can be easily verified that the tree construction ordering, $h_1 = AC,\ h_2 = CDF,\ h_3 = DEF,\ h_4 = FH,\ h_5 = BDE,\ h_6 = EFG$, together with the branching function $j(2) = 1,\ j(3) = 2,\ j(4) = 2,\ j(5) = 3,\ j(6) = 3$, defines the junction tree in Figure 3 (ii). The same ordering with a different branching function, namely, $j(2) = 1,\ j(3) = 2,\ j(4) = 3,\ j(5) = 3,\ j(6) = 3$, defines a different junction tree in Figure 3 (iii). Consider a different tree construction ordering, i.e., $h_1 = AC,\ h_2 = CDF,\ h_3 = DEF,\ h_4 = BDE,\ h_5 = EFG,\ h_6 = FH$, together with the branching function $j(2) = 1,\ j(3) = 2,\ j(4) = 3,\ j(5) = 3,\ j(6) = 5$, it defines the junction tree in Figure 3 (iv). It can also be easily verified that the separator sets of those junction trees in Figure 3 (ii), (iii), and (iv) are not only the same, but also identical to the separator set of the hypertree $\mathcal{H}$ in Figure 3 (i).

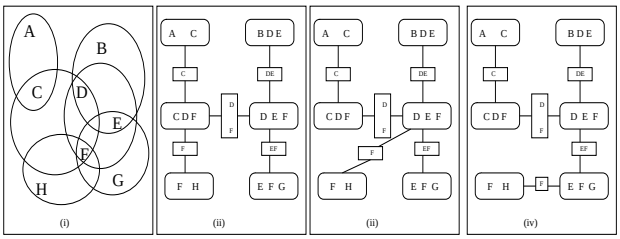

Figure 3: (i) The hypertree $\mathcal{H}$. (ii), (iii), (iv) Three possible junction tree representations of $\mathcal{H}$ in (i), respectively.

3. BNs and RDBs

It has been noticed that there exists a strong relationship between BNs and RDBs. There are two previously obtained results that contribute to the new proposed algorithm. (1) The relationship between full CIs and MVDs. (2) Constructing an acyclic database scheme.

Let R be a finite set of symbols, called *attributes*. We define a *database scheme* $\mathbf{R} = \{R_1,\ R_2,\ \ldots,\ R_n\}$ to be a set of subsets of R, where $R = R_1 \cup \ldots \cup R_n$. Each R_i in $\mathbf{R}$ is a *relation scheme*. A relation defined over a scheme R_i is denoted $r[R_i]$. A database scheme $\mathbf{R}$ can be treated as if it is a hypergraph each of whose hyperedges is one of the relation schemes in $\mathbf{R}$. On the other hand, for a hypergraph, we can treat each of its hyperedge as a relation scheme and all its hyperedges constitute a database scheme. A database scheme $\mathbf{R}$ is *acyclic* if its corresponding hypergraph is acyclic (Maier 1983).

Multivalued Dependence (MVD) is an important class of data dependence that has been intensively studied in RDBs.

Definition 10 A relation $r[R]$ is said to satisfy the (full) *multivalued dependency* (MVD) (Maier 1983), denoted, $Y \rightarrow\rightarrow X|Z$, if $r[R] = r[XY] \bowtie r[YZ]$, where $\bowtie$ denotes the natural join operator in relational algebra, $R = XYZ$, $r[XY]$ and $r[YZ]$ are projections (Maier 1983) of $r[R]$ onto schemes XY and YZ, respectively. Y is called the *key* of this MVD.

Definition 11 Given a set M of MVDs, the left hand sides of the MVDs in M are the *keys* of M.

A MVD $Y \rightarrow\rightarrow X|Z$ is said to *split* a set W of attributes if $W \cap X \neq \emptyset$ and $W \cap Z \neq \emptyset$.

Definition 12 A set M of (full) MVDs defined over a set R of attributes is *conflict free* (Maier 1983) if (i) the keys of M are not split by any MVD in M, and (ii) M satisfies

the *intersection property* (Maier 1983), that is, if $ZY \rightarrow\rightarrow X|W$ and $ZW \rightarrow\rightarrow X|Y$ are in M, then $Z \rightarrow\rightarrow X|YW \in M$, where $XYZW = R$.

Recently, a thorough study (Wong, Butz, & Wu 2000) has revealed that MVDs and full CIs, even though in different domains, correspond exactly to each other such that their implication problems coincide. This coincidence implies that the notions of *keys* and *split* can be carried over to full CIs so that we can define conflict free for full CIs in the exact same fashion as we did for MVDs.

Definition 13 (Wong, Butz, & Wu 2000) A set C of full CIs defined over a set V of variables is *conflict free* if (i) the keys of C are not split by any CI in C, and (ii) C satisfies the *intersection property*, that is, if $ZY \Rightarrow\Rightarrow X|W$ and $ZW \Rightarrow\Rightarrow X|Y$ are in C, then $Z \Rightarrow\Rightarrow X|YW \in C$, where $XYZW = V$.

The significance of conflict free MVDs is that they can be used to construct an acyclic database scheme (a hypertree) (Maier 1983). An efficient algorithm (Lien 1982), referred to as *Lien-Algorithm-MVDs* in this paper, has been developed to construct a unique hypertree $\mathcal{H}$ from a set of conflict free MVDs.

4. The New Relational Database Method

In this section, we will present a new method for constructing the MPD-JT of a BN motivated by the CI information encoded in the MPD-JT. We first briefly review the existing graphical algorithm in (Olesen & Madsen 2002), referred to as *Olesen-Madsen-MPD-JT*, using an example. The observations revealed by the examination motivate the development of the new method.

The *Olesen-Madsen-MPD-JT* algorithm takes a DAG as input, moralizes and triangulates it. The algorithm then removes any redundant fill-in edges so that a minimal triangulation is obtained and a normal junction tree is constructed. After that, it checks each separator of the resulting junction tree to decide whether it needs to aggregate the incidental cliques connected by the separator.

Consider the "Asia" BN in Example 1. After constructing the junction tree $\mathcal{T}$, the algorithm picks a junction tree separator $S \in \mathcal{S}^{\mathcal{T}}$ and tests whether S induces a complete subgraph of $M^{\mathcal{D}}$ (i.e., test whether $M^{\mathcal{D}}(S)$ is a complete subgraph of $M^{\mathcal{D}}$). If $M^{\mathcal{D}}(S)$ is not a complete subgraph, then the incidental cliques of S, say, C and $C^{'}$ in the junction tree $\mathcal{T}$, will be aggregated to obtain a bigger node which is the union of C and $C^{'}$. For instance, when testing the separator $BL \in \mathcal{S}^{\mathcal{T}}$ in Figure 1 (iv), it can be verified that the subgraph induced by nodes B and L is not a complete subgraph of $M^{\mathcal{D}}$ because B and L are not connected in the moralized graph $M^{\mathcal{D}}$ as shown in Figure 1 (ii). Since BL is connecting cliques BLS and BEL, they have to be aggregated to obtain a bigger node, i.e, $BELS$, to replace the cliques BLS, BEL and the separator BL, which results in Figure 2 (iii). The new structure in Figure 2 (iii) is still a junction tree though. It is important to note that after the cliques BLS and BES have been aggregated, the separator BL has been eliminated. This process repeats until all the separators in $\mathcal{S}^{\mathcal{T}}$ have been examined. It can be easily verified that all the other separators in the separator set $\mathcal{S}^{\mathcal{T}}$ induce complete subgraphs so that there are no more aggregations. Therefore, the final resulting MPD-JT after all necessary aggregations, shown in Figure 2 (iii), keeps only those separators originally in $\mathcal{S}^{\mathcal{T}}$ which induce complete subgraphs of $M^{\mathcal{D}}$.

It is well known that for any junction tree, each of its separators induces a full CI satisfied by the JPD defined by the BN (Pearl 1988). More specifically, consider a junction tree $\mathcal{T}$ consisting of cliques $C_i,\ i = 1,\ \ldots, k$, with its separator set $\mathcal{S}^{\mathcal{T}}$. If we delete a separator $S \in \mathcal{S}^{\mathcal{T}}$ and its incidental edges from $\mathcal{T}$, $\mathcal{T}$ will be separated into two disconnected parts. Without loss of generality, assuming one part contains $C_1, \ldots, C_m$, the other part contains the rest, the CI induced by S is $S \Rightarrow\Rightarrow \bigcup_{i=1}^{m} C_i - S | \bigcup_{j=m+1}^{k} C_j - S$. Also note that each induced CI corresponds to a decomposition $(\bigcup_{i=1}^{m} C_i - S,\ S,\ \bigcup_{j=m+1}^{k} C_j - S)$ of the triangulated graph from which the junction tree $\mathcal{T}$ is constructed, with S being a minimal complete separator.

Example 4 Consider the MPD-JT $\mathcal{T}$ shown in Fig. 2 (iii) consisting of cliques $C_1 = BELS$, $C_2 = ELT$, $C_3 = BDE$, $C_4 = AT$, and $C_5 = EX$, with its separator set $\mathcal{S}^{\mathcal{T}} = \{EL,\ BE,\ T,\ E\}$. The CI induced by the separator EL is $EL \Rightarrow\Rightarrow C_1 \cup C_3 \cup C_5 - EL \mid C_2 \cup C_4 - EL$, or $EL \Rightarrow\Rightarrow AT|BDSX$. Similarly, the CIs induced by the separators BE, T and, E are $BE \Rightarrow\Rightarrow D|ALSTX$ and $BE \Rightarrow\Rightarrow X|ADLST$, $T \Rightarrow\Rightarrow A|BDELSX$, and $E \Rightarrow\Rightarrow X|ABDLST$, respectively.

Recall the *Lien-Algorithm-MVDs* algorithm which constructs an acyclic database scheme from a conflict free set of MVDs, because of the correspondence between full CIs and MVDs, this algorithm has been carried over to the domain of BNs. In (Wong & Butz 2001), an algorithm referred to as *Lien-Algorithm-CIs* in this paper, was designed to construct a unique hypertree $\mathcal{H}$ from a set C of conflict free full CIs [2]. Recall the fact that a junction tree has a corresponding hypertree and a junction tree can be derived from a hypertree, this turns the problem of constructing an MPD-JT into the problem of constructing a hypertree from which this MPD-JT can be derived. Furthermore, the problem of constructing a hypertree, according to the algorithm *Lien-Algorithm-CIs*, can be turned into the problem of obtaining a conflict free set of full CIs. We therefore propose an alternative algorithm for constructing MPD-JT.

We put our focus on how to obtain a set of conflict free full CIs that can be used to construct the hypertree that corresponds to an MPD-JT. In Example 4, we demonstrated how to obtain a set of full CIs from a junction tree. It was shown that the full CIs identified as in Example 4 are conflict free and they are responsible for constructing the hypertree that corresponds to the junction tree from which this set of full CIs is identified. In the context of MPD-JTs, if we can obtain a set of conflict free full CIs from an MPD-JT *without* actually constructing it in the first place, then we can use

[2] See reference (Wong & Butz 2001) for detailed discussions on *Lien-Algorithm-CIs* and examples.

this set as input to the algorithm *Lien-Algorithm-CIs* to construct a hypertree that corresponds to the MPD-JT we want to construct. The following theorem guarantees that such a conflict free set of CIs is obtainable.

Consider a DAG $\mathcal{D}$ and its MPD-JT $\mathcal{T}$, let C denote the set of conflict free full CIs induced by separators in $S^{\mathcal{T}}$.

Theorem 1 $Y \Rightarrow\Rightarrow X|Z \in C$ if and only if $(X,\ Y,\ Z)$ is a decomposition of $M^{\mathcal{D}}$ with Y being the minimal complete separator.

Proof 1 ($\Rightarrow$) If $Y \Rightarrow\Rightarrow X|Z \in C$, then $Y \in \mathcal{S}^{\mathcal{T}}$ is a minimal complete separator of the triangulated graph $G^{\mathcal{T}}$ (from which $\mathcal{T}$ is constructed) and $(X,\ Y,\ Z)$ is a decomposition of $G^{\mathcal{T}}$. If $(X,\ Y,\ Z)$ is a decomposition of $G^{\mathcal{T}}$, then it implies that for every path in $G^{\mathcal{T}}$ from $A_i \in X$ to $A_j \in Z$, this path must pass through at least one vertex in Y. However, since every path from A_i to A_j in the moralized graph $M^{\mathcal{D}}$ is also a path in $G^{\mathcal{T}}$, hence every path in $M^{\mathcal{D}}$ from X to Z passes through Y, therefore $(X,\ Y,\ Z)$ is a decomposition of $M^{\mathcal{D}}$ with Y being the minimal complete separator.

($\Leftarrow$) In (Wong, Wu, & Butz 2002), it was proved that the decomposition $(X,\ Y,\ Z)$ of $M^{\mathcal{D}}$ with Y being a minimal complete separator is also a decomposition in every triangulated graph of $M^{\mathcal{D}}$. Therefore, Y is a junction tree separator in any junction tree that may be produced from $M^{\mathcal{D}}$, including the MPD-JT $\mathcal{T}$. Since the algorithm *Olesen-Madsen-MPD-JT* only eliminates junction tree separators which do not induce complete subgraphs in $M^{\mathcal{D}}$, the decomposition $(X,\ Y,\ Z)$ will be kept in the final resulting MPD-JT. Therefore, $Y \Rightarrow\Rightarrow X|Z \in C$.

Theorem 1 implies that the set C of conflict free full CIs encoded in an MPD-JT can be obtained in advance from the moralized graph of the original given DAG without constructing the MPD-JT. More precisely, the set C can be obtained by examining every decomposition $(X,\ Y,\ Z)$ of the moralized graph such that Y is a minimal complete separator. After the set C is obtained, we can then use it as input to the algorithm *Lien-Algorithm-CIs* to construct a hypertree and afterwards the MPD-JT. This analysis naturally results in the following algorithm for constructing an MPD-JT.

Algorithm *Relational-MPD-JT*
Input: a Bayesian network with its DAG $\mathcal{D}$;
Output: an MPD-JT of $\mathcal{D}$.

Step 1. Moralize the DAG $\mathcal{D}$ to obtain $M^{\mathcal{D}}$.
Step 2. Identify each decomposition $(X,\ Y,\ Z)$ of $M^{\mathcal{D}}$ where Y is a minimal complete separator of $M^{\mathcal{D}}$, and put the CI $Y \Rightarrow\Rightarrow X|Z$ in a set C.
Step 3. Invoke the *Lien-Algorithm-CIs* with C as input to obtain a hypertree $\mathcal{H}$.
Step 4. Return a junction tree $\mathcal{T}$ constructed from $G^{\mathcal{H}}$.

Chronologically, prior to the algorithm *Olesen-Madsen-MPD-JT*, there existed another graphical algorithm (Lemer 1993), referred to as *Graphical-MPD*, which can find all the mp-subgraphs of an arbitrary undirected graph (without going through the auxiliary junction tree as *Olesen-Madsen-MPD-JT* did). The *Graphical-MPD* algorithm was designed as a pure graphical procedure which has applications in some graph problems and has the time complexity of $\mathrm{O}(ne)$, where n is the number of vertices and e is the number of edges in the undirected graph.

The *Olesen-Madsen-MPD-JT* algorithm constructs the MPD-JT in the context of BNs via a junction tree obtained by minimal triangulation. This algorithm is also a graphical algorithm and specifically designed for BNs. The worst case complexity of this algorithm is dominated by triangulating the moralized graph which has the time complexity $\mathrm{O}(ne)$ while the time complexity of all other steps are no worse than this (Olesen & Madsen 2002). In other words, the algorithm *Olesen-Madsen-MPD-JT* does not exhibit any improvement over the algorithm *Graphical-MPD*, as far as the complexity is concerned. Nevertheless, it provides a different graphical perspective for constructing an MPD-JT in the domain of BNs, namely, making full use of the intermediate junction tree obtained instead of directly working on the underlying moralized graph.

Consider the new algorithm *Relational-MPD-JT*, the complexity of moralizing the input DAG to obtain $M^{\mathcal{D}}$ in step 1 is $\mathrm{O}(n^2)$ (Olesen & Madsen 2002). The complexity of identifying minimal complete separator decompositions in $M^{\mathcal{D}}$ to form C in step 2 is $\mathrm{O}(ne)$ (Lemer 1993). Constructing the hypertree $\mathcal{H}$ from C in step 3 takes $\mathrm{O}(|C| \cdot n)$. Finally, constructing the MPD-JT $\mathcal{T}$ from $\mathcal{H}$ in step 4 takes $\mathrm{O}(n^2)$ to finish (Olesen & Madsen 2002). Again, the time complexity of the whole algorithm is dominated by step 2, i.e., $\mathrm{O}(ne)$. In other words, it has the same worst case time complexity with the *Graphical-MPD* and *Olesen-Madsen-MPD-JT* algorithms. However, we want to emphasize that the new proposed algorithm makes the following two contributions: (1) It provides a semantic explanation for the graphical construction of the MPD-JT in (Lemer 1993; Olesen & Madsen 2002) based on the information encoded in an MPD-JT. That is, constructing an MPD-JT is equivalent to obtaining a set of conflict full CIs from the moralized graph of a given DAG. (2) Since the notion of conflict free CIs is inherited and adapted from the notion of conflict free MVDs in RDBs, the new proposed algorithm makes full use of the existing algorithm *Lien-Algorithm-CIs*, which was inherited from the database algorithm *Lien-Algorithm-MVDs*, to construct an MPD-JT. In other words, the core construction step of the algorithm *Relational-MPD-JT* is an application of the result on constructing a hypertree in database theory being applied in the BN domain. This revelation farther confirms the intriguing relationship between BNs and RDBs.

5. Discussion

The bottleneck of the algorithm *Olesen-Madsen-MPD-JT* lies in the triangulation step whose complexity dominates the whole algorithm. Even though the new proposed algorithm does not need triangulation step, it needs to identify all the decompositions with minimal complete separators. It is perhaps worth pursuing how to shift the bottleneck of triangulation to decomposition identification which may result in a more efficient (perhaps approximate) algorithm.

Although the new proposed algorithm is within the context of BNs, the idea of identifying and obtaining a conflict free set of full CIs from the moralized graph of a DAG can also be applied to an arbitrary undirected graph, not necessarily a moralized graph of a DAG. Therefore, the proposed method for constructing an MPD-JT is not confined to the domain of BNs.

The relationship betwen conflict free MVDs (CIs) and hypertrees suggests that any graphical problem of constructing a hypertree can be equivalently considered as the problem of how to obtain a set of conflict free set of MVDs (CIs). This provides a new (algebraic) perspective for solving the graphical problem of constructing hypertrees (junction trees) in future research. For instance, the proposed algorithm may be possibly applied to obtain a hierarchical Markov network representation of a BN (Wong, Butz, & Wu 2001).

Acknowledgment

The authors wish to thank the referees for their helpful and constructive criticism.

References

Huang, C., and Darwiche, A. 1996. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning* 15(3):225–263.

Lauritzen, S., and Spiegelhalter, D. 1988. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society* 50:157–244.

Lemer, H. 1993. Optimal decomposition by clique separators. *Discrete Mathematics* 113:99–123.

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

Madsen, A. L., and Jensen, F. V. 1998. Lazy propagation in junction trees. In Cooper, G. F., and Moral, S., eds., *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 362–369. San Francisco: Morgan Kaufmann.

Maier, D. 1983. *The Theory of Relational Databases*. Rockville, Maryland: Computer Science Press.

Olesen, K., and Madsen, A. 2002. Maximal prime subgraph decomposition of bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics* 32(1):21–31.

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

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

Wen, W. 1991. From relational databases to belief networks. In *Seventh Conference on Uncertainty in Artificial Intelligence*, 406–413. Morgan Kaufmann Publishers.

Wong, S., and Butz, C. 2001. Constructing the dependency structure of a multi-agent probabilistic network. *IEEE Transactions on Knowledge and Data Engineering* 13(3):395–415.

Wong, S.; Butz, C.; and Wu, D. 2000. On the implication problem for probabilistic conditional independency. *IEEE Transactions on System, Man, Cybernetics, Part A: Systems and Humans* 30(6):785–805.

Wong, S.; Butz, C.; and Wu, D. 2001. On undirected representations of bayesian networks. In *ACM SIG/IR Workshop on Mathematical/Formal Models in Information Retrieval (MF/IR)*, 52–59.

Wong, S.; Wu, D.; and Butz, C. 2002. Triangulation of bayesian networks: a relational database perspective. In *The Third International Conference on Rough Sets and Current Trends in Computing (RSCTC 2002), LNAI 2475*, 389–397.