KYBERNETIKA — VOLUME *48* (2012), NUMBER 5, PAGES 825–844

A BACKWARD SELECTION PROCEDURE FOR APPROXIMATING A DISCRETE PROBABILITY DISTRIBUTION BY DECOMPOSABLE MODELS

Francesco M. Malvestuto

Decomposable (probabilistic) models are log-linear models generated by acyclic hypergraphs, and a number of nice properties enjoyed by them are known. In many applications the following selection problem naturally arises: given a probability distribution p over a finite set V of n discrete variables and a positive integer k, find a decomposable model with tree-width k that best fits p. If $\mathcal{H}$ is the generating hypergraph of a decomposable model and $p_{\mathcal{H}}$ is the estimate of p under the model, we can measure the closeness of $p_{\mathcal{H}}$ to p by the information divergence $D(p : p_{\mathcal{H}})$, so that the problem above reads: given p and k, find an acyclic, connected hypergraph $\mathcal{H}$ of tree-width k such that $D(p : p_{\mathcal{H}})$ is minimum. It is well-known that this problem is NP-hard. However, for $k = 1$ it was solved by Chow and Liu in a very efficient way; thus, starting from an optimal Chow–Liu solution, a few forward-selection procedures have been proposed with the aim at finding a 'good' solution for an arbitrary k. We propose a backward-selection procedure which starts from the (trivial) optimal solution for $k = n - 1$, and we show that, in a study case taken from literature, our procedure succeeds in finding an optimal solution for every k.

Keywords: backward selection, information divergence, decomposable model, acyclic hypergraph, k-hypertree

Classification: 05C65, 62-09, 68R10, 68T05

1. INTRODUCTION

An important area of machine learning research is the development and use of probabilistic models for classification and prediction. A class of popular probabilistic models is that of "decomposable models" [8, 26] (also called "multiplicative models" in [42, 43]), which enjoy a number of nice properties: they are not tied to a specific form of a distribution, their interpretations are relatively easy because one can distinguish variables that belong together from variables that can be separated, no iterative fitting algorithms are needed to compute estimates, they can be characterized by zero partial associations (that is, conditional independences). Decomposable models are loglinear models generated by "acyclic hypergraphs". A hypergraph $\mathcal{H}$ is *acyclic* (or "decomposable") if there exists a *running-intersection ordering* of $\mathcal{H}$, that is, an ordering $(A_1, \dots, A_m)$ of its hyperedges such that, if $m > 1$ then, for each $i \geq 2$, there exists $j < i$ for which

$$(A_1 \cup \ldots \cup A_{i-1}) \cap A_i \subseteq A_j.$$

If $\mathcal{H}$ is a reduced and connected hypergraph with $m > 1$ hyperedges (for details see Section 2), then the sets $(A_1 \cup \ldots \cup A_{i-1}) \cap A_i$, $2 \leq i \leq m$, are called the *separators* of $\mathcal{H}$ [26].

In some applications, one or more decomposable models are assigned in advance and one wants to find all the decomposable models that can be logically inferred [18, 19, 23, 30, 31]. But in other applications, we are given a sample of observations of n categorical variables, and we want to select a decomposable model that best fits the (probability) distribution p defined by frequencies of events in the sample, subject to a constraint owing to the allowable amount of machine memory [9, 25, 27]. Given the generating hypergraph $\mathcal{H}$ of a decomposable model, by $p_{\mathcal{H}}$ we denote the estimate of p under the model, and we measure the accuracy of the estimate using the information divergence $D(p : p_{\mathcal{H}})$, which is a nonnegative quantity vanishing if and only if $p_{\mathcal{H}} = p$. Let k be the *tree-width* of $\mathcal{H}$, that is, k is the maximum cardinality of hyperedges of $\mathcal{H}$ minus one. Then, the selection problem above is reducible to the following parameterized problem:

(P) Given a distribution p over a set V of n discrete variables and an integer k, $0 \leq k \leq n-1$, find an acyclic hypergraph $\mathcal{H}$ on V with tree-width k such that $D(p : p_{\mathcal{H}})$ is minimum.

We call an acyclic hypergraph on V with tree-width k a *k-solution*, and a k-solution $\mathcal{H}$ is *optimal* if $D(p : p_{\mathcal{H}})$ is minimum. Note that the point partition of V, that is, the hypergraph $\{\{v\} : v \in V\}$, is the only 0-solution and, hence, it is the optimal 0-solution; moreover, the trivial partition of V, that is, the hypergraph $\{V\}$, is the only $(n-1)$-solution and, hence, it is the optimal $(n-1)$-solution. So, problem (P) has a trivial optimal k-solution for $k = 0$ and $k = n-1$. Moreover, Chow and Liu [11] showed that a connected 1-solution with minimum information divergence (a "Chow–Liu tree") can be found in a very efficient way. However, the problem of finding an optimal k-solution for $1 < k < n-1$ was proven to be NP-hard [10, 14, 15, 21, 35], and several heuristics have been proposed to find an approximation to an optimal k-solution. Using the "forward approach" [2, 3, 4, 16, 17, 37, 39, 44], one starts from a Chow–Liu tree and adds edges while maintaining chordality [16] and keeping the tree-width non greater than k; with an appropriate choice of edges to add, the set of maximal cliques of the resultant chordal graph is a 'good' approximation to an optimal k-solution. Another approach is based on the following result.

Theorem 1. (Malvestuto [29]) For every acyclic hypergraph $\mathcal{H}$ on V with tree-width k, there exists an acyclic hypergraph $\mathcal{H}^*$ on V such that

(H1) $\mathcal{H}^*$ is connected,

(H2) every hyperedge of $\mathcal{H}^*$ has cardinality $k+1$,

(H3) every separator of $\mathcal{H}^*$ has cardinality k,

and for which $D(p : p_{\mathcal{H}^*}) \leq D(p : p_{\mathcal{H}})$.

We shall see in Section 3 that an acyclic hypergraph meeting requirements (H1), (H2) and (H3) can be viewed as being the set of maximal cliques of a "k-tree" [7, 22, 38] and, henceforth, such hypergraphs will be referred to as *k-hypertrees.* Theorem 1 has recently been re-discovered in [39], where k-hypertrees are called "t-cherry junction trees".

By Theorem 1, there always exists a k-hypertree that is an optimal k-solution. In [28, 29] and in [40] the authors gave two procedures which construct a k-hypertree by adding $(k+1)$-sets incrementally in such a way that the output is an approximation to an optimal k-solution.

Finally, a "backward approach" was theorized by Wermuth [42, 43], who provided a criterion for decomposing a hyperedge A of an acyclic hypergraph $\mathcal{H}$ into two subsets $A-\{u\}$ and $A-\{v\}$ such that the hypergraph $(\mathcal{H}-\{A\})\cup\{A-\{u\}, A-\{v\}\}$ is acyclic. Recently, a backward-selection procedure has been worked out in [36] based on a minimal triangulation algorithm, and the authors tested their algorithm on the *Jamaican Lizards* dataset [8] which we will examine in Section 5.

In this paper, we present a backward algorithm (see the `Greedy Backward` algorithm in Section 4) which constructs a suitable $(h-1)$-hypertree from an h-hypertree. Thus, if we start with the the optimal $(n-1)$-solution (that is, the trivial partition of V) and apply our algorithm $n-k-1$ times (with $h=n-1,\dots,k+1$), then we find an approximation to an optimal k-solution. We will test our procedure on the *Jamaican Lizards* dataset and show that it succeeds in finding an optimal k-solution for each k.

The paper is organized as follows. Section 2 contains basic definitions and preliminary results on acyclic hypergraphs, decomposable models and estimates of a probability distribution. In Section 3 we introduce the notions of a k-hypertree and of an optimal k-hypertree, and state some preliminary results. In Section 4 we present our greedy backward algorithm, which is tested in Section 5 on the *Jamaican Lizards* dataset. Section 6 contains a closing note.

2. PRELIMINARIES

2.1. Acyclic hyperhgraphs

We start with some more-or-less standard definitions. A *hypergraph* on a finite nonempty set V is a set $\mathcal{H}$ of subsets of V whose union recovers V; the elements of $\mathcal{H}$ and V are called the *hyperedges* (the *edges*, for short) of $\mathcal{H}$ and the *vertices* of $\mathcal{H}$, respectively. If $\mathcal{H}$ is the trivial partition of V (that is, $\mathcal{H}=\{V\}$), we call $\mathcal{H}$ the *trivial hypergraph* on V. Two vertices of $\mathcal{H}$ are *adjacent* if they belong both to some edge of $\mathcal{H}$. A *clique* of $\mathcal{H}$ is a nonempty subset C of V such that either $|C|=1$ or every two distinct vertices in C are adjacent. A *k-clique* of $\mathcal{H}$ is a clique of $\mathcal{H}$ with k vertices. A hypergraph is *connected* if every two distinct vertices a and b are joined by a path, that is, there is a sequence of distinct vertices $(v_1,\dots,v_k)$ such that $v_1=a$, $v_k=b$ and, for $h=1,\dots,k-1$, the vertices v_h and v_{h+1} are adjacent. A hypergraph is *reduced* if no edge is a subset of another edge.

Henceforth, we limit our considerations to hypergraphs that are connected and reduced.

An (undirected, simple) *graph* is a hypergraph whose edges have all cardinality 2. The *adjacency graph* (or "2-section") of a hypergraph $\mathcal{H}$ on V is the graph on V, denoted by $\mathcal{H}_2$, where two vertices are adjacent if and only if they adjacent in $\mathcal{H}$.

Recall from the Introduction that a hypergraph is acyclic if there exists a running-intersection ordering of its edges. A linear algorithm to find a running-intersection ordering (if any) of a hypergraph was given in [41]. Several equivalent definitions of acyclicity exist [6]. Let $\mathcal{H}$ be a nontrivial acyclic (connected and reduced) hypergraph with m edges, let $(A_1, \ldots, A_m)$ be a running-intersection ordering of $\mathcal{H}$, and let $B_i = (A_1 \cup \ldots \cup A_{i-1}) \cap A_i$, $2 \le i \le m$. The sets $B_2, \ldots, B_m$ are precisely the (*minimal vertex*) *separators* of $\mathcal{H}$ [23]. For each separator B of $\mathcal{H}$, the *replication number* of B, denoted by r_B, is the number of distinct values of the index i for which $B = B_i$.

Fact 1. The replication number of a separator of $\mathcal{H}$ is the same for every running-intersection ordering of $\mathcal{H}$.

Remark 1. Let $(A_1, \ldots, A_m)$ be a running-intersection ordering of an acyclic nontrivial hypergraph $\mathcal{H}$, and let $B_i = (A_1 \cup \ldots \cup A_{i-1}) \cap A_i$, $2 \le i \le m$. Since every two vertices in A_1 are adjacent and, for each i, $2 \le i \le m$, every two vertices in $A_i - B_i$ are adjacent and each vertex in $A_i - B_i$ is adjacent to every vertex in B_i, the number of edges of $\mathcal{H}_2$ is exactly

$$\binom{|A_1|}{2} + \sum_{i=2,\ldots,m} \alpha_i$$

where

$$\alpha_i = \begin{cases} |B_i| & \text{if } |A_i - B_i| = 1 \\ \binom{|A_i - B_i|}{2} + |A_i - B_i| \cdot |B_i| & \text{else.} \end{cases}$$

Acyclic (connected) hypergraphs can be represented by trees (e. g., "junction trees" [20, 26], "join trees" [6] and "Almond trees" [1, 20]). We shall make use of a tree-representation of an acyclic (connected and reduced) hypergraph, called "edge-separator tree" (also called "edge-divider tree" [5] and "connection tree" [34]), which is halfway between a junction tree and an Almond tree. Let $\mathcal{H}$ be an acyclic hypergraph on V, and let $\mathcal{S}$ be the set of separators of $\mathcal{H}$. An *edge-separator tree* of $\mathcal{H}$ is an undirected tree T with node set $\mathcal{H} \cup \mathcal{S}$ that satisfies the following property (called "separation property" in [1] and "junction property" in [26]): For every vertex v of $\mathcal{H}$, the subgraph of T induced by those nodes that contain v is connected. Note that, in order to avoid ambiguities, we call *nodes* and *arcs* the vertices and edges of T, respectively; moreover, we call a node of T an *edge-node* (or a *separator-node*) if it belongs to $\mathcal{H}$ (to $\mathcal{S}$, respectively). Observe that, if $\mathcal{H}$ is the trivial hypergraph (that is, $\mathcal{H} = \{V\}$), then $\mathcal{S} = \emptyset$ so that the edge-separator tree of $\mathcal{H}$ is a one-point tree whose unique node is V.

Fact 2. Let T be any edge-separator tree of an acyclic hypergraph. The number of neighbors of a separator-node B of T is equal to $r_B + 1$.

From a computational point of view, the set $\mathcal{S}$ of separators of an acyclic hypergraph $\mathcal{H}$ can be obtained in time linear in $||\mathcal{H}|| = \sum_{A \in \mathcal{H}} |A|$ [41], and an edge-separator tree of $\mathcal{H}$ can be constructed in time linear in $||\mathcal{H}|| \cdot ||\mathcal{S}||$ where $||\mathcal{S}|| = \sum_{B \in \mathcal{S}} |B|$ [5].

In all the examples that follow, we write sets of vertices simply as lists.

Example 1. Consider the acyclic hypergraph $\mathcal{H} = \{ae, be, cd, ce\}$. The set of separators of $\mathcal{H}$ is $\mathcal{S} = \{c, e\}$; moreover, one has $r_c = 1$ and $r_e = 2$. The hypergraph has exactly one edge-separator tree which is shown in Figure 1.

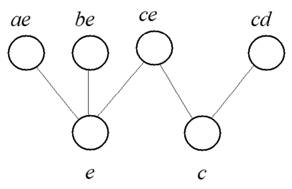

Fig. 1. The edge-separator tree of the acyclic hypergraph $\{ae, be, cd, ce\}$.

A *rooted edge-separator tree* of $\mathcal{H}$ is an edge-separator tree of $\mathcal{H}$ rooted at any edge-node; if R is the root, the rooted edge-separator tree is denoted by T_R. For every two adjacent nodes N and N' of T_R, N is the *parent* of N' (or, equivalently, N' is a *child* of N) if the distance of N from R is less than the distance of N' from R. A *leaf* of T_R is an (edge-)node with no children. It is worth noting that, given a rooted edge-separator tree T_R of $\mathcal{H}$, we can construct a running-intersection ordering of $\mathcal{H}$, by visiting the nodes of T_R in a top-down way, that is, by visiting a node after its parent.

2.2. Decomposable probabilistic models

Let V be a finite set of discrete (random) variables. Let A be a nonempty (proper or improper) subset of V; an A-*tuple* is an assignment of values to variables in A. Given a V-tuple x and a nonempty subset A of V, by x_A we denote the A-tuple obtained from x by ignoring the values of variables in $V - A$.

Let p be a (probability) distribution over V; for V-tuple x, by $p(x)$ we denote the value of p at x. Let A be a nonempty subset of V; by p_A we denote the *marginal* of p on A so that for A-tuple y one has $p_A(y) = \sum_{x:x_A=y} p(x)$, and by $E_p(A)$ we denote the *entropy* of p_A, that is,

$$E_p(A) = \sum_{y:p_A(y)>0} p_A(y) \log p_A(y).$$

Let A be a subset of V with $|A| \geq 2$, and let u and v be two distinct variables in A. By $I_p(u, v|A - \{u, v\})$ we denote the *conditional mutual information* [12, 13] (also called "average conditional information" [29]) of u and v *given* $A - \{u, v\}$, that is,

$$I_p(u, v|A - \{u, v\}) = E_p(A - \{u\}) + E_p(A - \{v\}) - E_p(A - \{u, v\}) - E_p(A)$$

where, by convention, $E_p(A - \{u, v\})$ is taken to be 0 if $A = \{u, v\}$. It is well-known [12, 13] that $I_p(u, v|A - \{u, v\}) \geq 0$, where the equality holds if and only if u and v are conditionally independent given $A - \{u, v\}$ under p_A.

A distribution q over V *dominates* p if $q(x) > 0$ for every V-tuple x for which $p(x) > 0$. If q dominates p, the *information divergence* (or "Kullback-Leibler distance" or "cross-entropy"or "relative entropy") of q from p is defined by the quantity

$$D(p:q) = \sum_{x:p(x)>0} p(x) \log \frac{p(x)}{q(x)}.$$

It is well-known that $D(p:q) \geq 0$ where the equality holds if and only if $p = q$.

The *log-linear model* [26] generated by a hypergraph $\mathcal{H}$ on V is the set of distributions over V that factorize according to $\mathcal{H}$. A (*probabilistic*) *decomposable model* [26, 32] is a log-linear model generated by an acyclic hypergraph. A distribution is *decomposable* by an acyclic hypergraph $\mathcal{H}$ if it belongs to the decomposable model generated by $\mathcal{H}$.

Proposition 1. [27] Let $\mathcal{H}$ be an acyclic hypergraph on V, and let $\mathcal{S}$ be the set of separators of $\mathcal{H}$. A distribution p is decomposable by $\mathcal{H}$ if and only if

(i) for every V-tuple x with $p(x) > 0$ one has $p(x) = \dfrac{\prod_{A\in\mathcal{H}} p_A(x_A)}{\prod_{B\in\mathcal{S}} (p_B(x_B))^{r_B}}$, and

(ii) for every V-tuple x with $p(x) = 0$ there exists an edge A of $\mathcal{H}$ for which $p_A(x_A) = 0$.

In what follows, we summarize the two conditions of Proposition 1 simply by writing

$$p = \frac{\prod_{A\in\mathcal{H}} p_A}{\prod_{B\in\mathcal{S}} p_B^{r_B}}$$

Equivalenty, given a running-intersection ordering $(A_1, \ldots, A_m)$ of $\mathcal{H}$, $m \geq 2$, p is decomposable by $\mathcal{H}$ if and only if

$$p_{\mathcal{H}} = p_{A_1} \prod_{2\leq i\leq n-k} \frac{p_{A_i}}{p_{B_i}}$$

where $B_i = (A_1 \cup \ldots \cup A_{i-1}) \cap A_i$, $2 \leq i \leq m$.

Example 1 (continued). By Proposition 1, a distribution p over the variable set *abcde* is decomposable by $\mathcal{H} = \{ae, be, cd, ce\}$ if and only if $p = \dfrac{p_{ae}p_{be}p_{cd}p_{ce}}{p_c p_e^2}$.

Let p be a distribution over V and let $\mathcal{H}$ be a hypergraph on V. By $M_{\mathcal{H}}(p)$ we denote the set of the marginals of p over $\mathcal{H}$, that is, $M_{\mathcal{H}}(p) = \{p_A : A \in \mathcal{H}\}$. The *estimate* of p under the log-linear model generated by $\mathcal{H}$, denoted by $p_{\mathcal{H}}$, is the *maximum-entropy extension* of $M_{\mathcal{H}}(p)$.

Let $\mathcal{H}$ be an acyclic hypergraph on V, and let $\mathcal{S}$ be the set of separators of $\mathcal{H}$. Then $p_{\mathcal{H}}$ factorizes as follows [26, 32]:

$$p_{\mathcal{H}} = \frac{\prod_{A\in\mathcal{H}} p_A}{\prod_{B\in\mathcal{S}} p_B^{r_B}}. \tag{1}$$

Note that, since $(p_{\mathcal{H}})_A = p_A$ for every edge A of $\mathcal{H}$, $p_{\mathcal{H}}$ is decomposable by $\mathcal{H}$ by Proposition 1. Moreover, the entropy of $p_{\mathcal{H}}$ amounts to

$$E_{p_{\mathcal{H}}}(V) = \sum_{A \in \mathcal{H}} E_p(A) - \sum_{B \in \mathcal{S}} r_B E_p(B).$$

Finally, it is easy to see that $p_{\mathcal{H}}$ dominates p: if $p(x) > 0$ then $p_A(x_A) > 0$ for each edge A of $\mathcal{H}$ and $p_B(x_B) > 0$ for each separator B of $\mathcal{H}$ so that, by (1), $p_{\mathcal{H}}(x) > 0$. Therefore, the information divergence $D(p : p_{\mathcal{H}})$ is well-defined and, explicitly, one has [29, 40]

$$D(p : p_{\mathcal{H}}) = E_{p_{\mathcal{H}}}(V) - E_p(V). \tag{2}$$

The next result involves two acyclic hypergraphs $\mathcal{H}$ and $\mathcal{G}$ on the same vertex set where $\mathcal{G}$ is *finer* than $\mathcal{H}$, by which we mean that every edge of $\mathcal{G}$ is a subset of some edge of $\mathcal{H}$.

Theorem 2. Let p be a distribution over V, and let $\mathcal{H}$ and $\mathcal{G}$ be two acyclic hypergraphs on V. If $\mathcal{G}$ is finer than $\mathcal{H}$ then

(*i*) $E_{p_{\mathcal{H}}}(V) \leq E_{p_{\mathcal{G}}}(V)$;

(*ii*) $p_{\mathcal{G}}$ dominates $p_{\mathcal{H}}$ and $D(p_{\mathcal{H}} : p_{\mathcal{G}}) = D(p : p_{\mathcal{G}}) - D(p : p_{\mathcal{H}})$.

Proof. (*i*) Since $\mathcal{G}$ is finer than $\mathcal{H}$, every extension of $M_{\mathcal{H}}(p)$ is an extension of $M_{\mathcal{G}}(p)$ so that $p_{\mathcal{H}}$ is an extension of $M_{\mathcal{G}}(p)$ and, hence, $E_{p_{\mathcal{H}}}(V) \leq E_{p_{\mathcal{G}}}(V)$.
(*ii*) Let x be a V-tuple with $p_{\mathcal{H}}(x) > 0$. By eq. (1), one has that $p_A(x_A) > 0$ for every edge A of $\mathcal{H}$. Let A' be any edge of $\mathcal{G}$. Since $\mathcal{G}$ is finer than $\mathcal{H}$, there exists an edge of $\mathcal{H}$ that contains A'. Let A be such an edge of $\mathcal{H}$; then, one has $p_{A'}(x_{A'}) \geq p_A(x_A) > 0$. Since this is true for every edge A' of $\mathcal{G}$, by eq. (1), one has that $p_{\mathcal{G}}(x) > 0$, which proves that $p_{\mathcal{G}}$ dominates $p_{\mathcal{H}}$. So, $D(p_{\mathcal{H}} : p_{\mathcal{G}})$ is well-defined. Moreover, by eq. (2), one has $D(p_{\mathcal{H}} : p_{\mathcal{G}}) = E_{p_{\mathcal{G}}}(V) - E_{p_{\mathcal{H}}}(V)$. On the other hand, again by eq. (2), one has $D(p : p_{\mathcal{H}}) = E_{p_{\mathcal{H}}}(V) - E_p(V)$ and $D(p : p_{\mathcal{G}}) = E_{p_{\mathcal{G}}}(V) - E_p(V)$. To sum up, one has $D(p_{\mathcal{H}} : p_{\mathcal{G}}) = D(p : p_{\mathcal{G}}) - D(p : p_{\mathcal{H}})$. □

Finally, by eq. (1), the set of marginals $M_{\mathcal{H}}(p)$ is a convenient storage representation for $p_{\mathcal{H}}$ so that we can measure the *storage cost* of $p_{\mathcal{H}}$ by the *size* of $M_{\mathcal{H}}(p)$, by which we mean the amount of memory needed to store the marginals p_A for $A \in \mathcal{H}$.

3. ESTIMATES BY K-HYPERTREES

3.1. k-hypertrees

A graph is *complete* if its vertex set is a clique. A *k-tree* [7, 22] is a graph which can be formed by starting with a complete graph on $k+1$ vertices and, then, repeatedly adding vertices in such a way that each added vertex has exactly k neighbours that form a clique. It is easily seen that a k-tree with n vertices has $n-k$ maximal cliques.

By a *k-hypertree* we mean a hypergraph whose edges are maximal cliques of a k-tree. Accordingly, a k-hypertree is a hypergraph which can be formed by starting with a trivial hypergraph on $k+1$ vertices and, then, repeatedly adding edges of the type $C \cup \{a\}$ where C is a k-clique (of the already constructed k-hypertree) and a is a new vertex. Note that the 0-hypertree on V is the point partition of V, and a 1-hypertree on V is an ordinary tree on V. Since a k-tree with n vertices has $n-k$ maximal cliques, a k-hypertree with n vertices has exactly $n-k$ edges; moreover, by Remark 1, the adjacency graph of a k-hypertree on n vertices is a k-tree with $\left(n - \frac{k+1}{2}\right) k$ edges.

It is easy to show that a k-hypertree $\mathcal{H}$ with n vertices has exactly one edge-separator tree. The statement is obvious if $k = n-1$. Assume that $k < n-1$ and let T and T' be two edge-separator trees of $\mathcal{H}$. In order to prove that $T = T'$, it is sufficient to show that, if A and A' are two edges of $\mathcal{H}$ that in T are joined by a path of length 2, then A and A' are joined by a path of length 2 in T', too. Let $\pi = (A, B, A')$ and $\pi' = (A, B_1, \ldots, B_l, A')$ be the paths joining A and A' in T and in T', respectively. Of course, one has $B = A \cap A'$. Moreover, by the junction property of T', the set $A \cap A'$ is contained in each node on π'; therefore, B $(= A \cap A')$ is a subset of each B_h, $1 \leq h \leq l$. Since the separators of $\mathcal{H}$ have all the same cardinality $(= k)$, one has $B = B_1 = \ldots = B_l$. It follows that $\pi = \pi'$ and, hence, $T = T'$.

Finally, it is a matter of course to verify that a hypergraph $\mathcal{H}$ is a k-hypertree if and only if $\mathcal{H}$ is an acyclic hypergraph meeting the three requirements (H1), (H2) and (H3) mentioned in the Introduction.

3.2. Optimal k-hypertrees

As stated in the Introduction, there always exists a k-hypertree on V that is an optimal k-solution to problem (P); we call such a k-hypertree an *optimal k-hypertree* for p. The next result shows that, for estimates of p generated by optimal k-hypertrees, both accuracy and storage cost are non-decreasing functions of k. Henceforth, we assume that each variable takes on the same number of values.

Theorem 3. Let $\mathcal{H}$ be an optimal k-hypertree for p and let $\mathcal{G}$ be an optimal $(k-1)$-hypertree for p. Then

(*i*) $D(p : p_{\mathcal{H}}) \leq D(p : p_{\mathcal{G}})$;

(*ii*) the size of $M_{\mathcal{H}}(p)$ is greater than or equal to the size of $M_{\mathcal{G}}(p)$.

Proof. (*i*) Let A and A' be two edges of $\mathcal{G}$ such that $A \cap A'$ equals a separator of $\mathcal{G}$. The hypergraph $\mathcal{G}'$ obtained from $\mathcal{G}$ by replacing the two edges A and A' with the set $A \cup A'$ is an acyclic hypergraph with tree-width k. Since $\mathcal{G}$ is finer than $\mathcal{G}'$, by Theorem 1 one has $D(p : p_{\mathcal{G}'}) \leq D(p : p_{\mathcal{G}})$. By Theorem 2, there exists a k-hypertree $\mathcal{H}'$ such that $D(p : p_{\mathcal{H}'}) \leq D(p : p_{\mathcal{G}'})$. Since $\mathcal{H}$ is an optimal k-hypertree for p, one also has $D(p : p_{\mathcal{H}}) \leq D(p : p_{\mathcal{H}'})$. To sum up, one has $D(p : p_{\mathcal{H}}) \leq D(p : p_{\mathcal{H}'}) \leq D(p : p_{\mathcal{G}'}) \leq D(p : p_{\mathcal{G}})$ which proves the statement.

(*ii*) Let us assume that each variable takes on d values. Since $\mathcal{H}$ has $n-k$ edges and each edge of $\mathcal{H}$ has cardinality $k+1$, the size of each marginal in $M_{\mathcal{H}}(p)$ is d^{k+1} and the size of $M_{\mathcal{H}}(p)$ is $(n-k)d^{k+1}$. Analogously, the size of $M_{\mathcal{G}}(p)$ is $(n-k+1)d^k$. Therefore, the size of $M_{\mathcal{H}}(p)$ minus the size of $M_{\mathcal{G}}(p)$ is $d^k[(d-1)(n-k)-1]$, which is not less than d^k as $d > 1$ and $n > k$. $\square$

4. A GREEDY BACKWARD ALGORITHM

Let $\mathcal{H}$ be a k-hypertree on a set V of n vertices, where $2 \leq k \leq n-1$. We first present a method which, given a running-intersection ordering $(A_1, \ldots, A_{n-k})$ of $\mathcal{H}$, constructs a $(k-1)$-hypertree $\mathcal{G}$ on V. The idea is that the $n-k+1$ edges of $\mathcal{G}$ can be obtained as follows: two edges of $\mathcal{G}$ are the pieces of any two-way decomposition of A_1 and, if $k < n-1$, then each of the remaining $n-k-1$ edges of $\mathcal{G}$ is a k-element subset of A_i, $i = 2, \ldots, n-k$. More precisely, we will construct a chain of hypergraphs $\mathcal{G}_1 \subset \ldots \subset \mathcal{G}_{n-k} = \mathcal{G}$ as follows:

— $\mathcal{G}_1 = \{A_1 - \{a_1\}, A_1 - \{b_1\}\}$ where (a_1, b_1) is any couple of elements of A_1;

— if $k < n-1$ then $\mathcal{G}_i$, $2 \leq i \leq n-k$, is obtained from $\mathcal{G}_{i-1}$ by adding one edge of the form $C_i \cup \{a_i\}$, where C_i is a $(k-1)$-element subset of A_i that is a clique of $\mathcal{G}_{i-1}$ and a_i is the unique element of the singleton $A_i - (A_1 \cup \ldots \cup A_{i-1})$.

First of all, we prove that, if $k < n-1$ then, for each i, $2 \leq i \leq n-k$, a clique of $\mathcal{G}_{i-1}$ such as C_i always exists. To see it, let $B_i = (A_1 \cup \ldots \cup A_{i-1}) \cap A_i$, and let $j_i < i$ be such that $B_i \subset A_{j_i}$, $2 \leq i \leq n-k$. Let us distinguish two cases depending on whether $j_i = 1$ or $j_i > 1$. If $j_i = 1$ then, since $|A_1| = k+1$ and $|B_i| = k$, one has that $a_1 \in B_i$ or $b_1 \in B_i$; therefore, $B_i - \{a_1\}$ or $B_i - \{b_1\}$ is a clique of $\mathcal{G}_1$ and, hence, of $\mathcal{G}_{i-1}$. Assume that $j_i > 1$. Let A' be the k-element subset of A_{j_i} that was added to $\mathcal{G}_{j_i - 1}$ to give $\mathcal{G}_{j_i}$. Then, the set $B_i \cap A'$ is a $(k-1)$-element subset of B_i that is a clique of $\mathcal{G}_{j_i}$ and, hence, of $\mathcal{G}_{i-1}$.

So, for each i, $1 \leq i \leq n-k$, $\mathcal{G}_i$ is a hypergraph on $V_i = \cup_{j=1,\ldots,i} A_j$. At this point, it is easy to prove by induction on i that each $\mathcal{G}_i$ is a $(k-1)$-hypertree on V_i. Trivially, $\mathcal{G}_1$ is a $(k-1)$-hypertree on V_1 $(= A_1)$. Note that the set of separators of $\mathcal{G}_1$ is the singleton $\mathcal{S}_1 = \{A_1 - \{a_1, b_1\}\}$. Moreover, if $\mathcal{G}_{i-1}$ is $(k-1)$-hypertree on V_{i-1} with separator set $\mathcal{S}_{i-1}$ then, since C_i is a clique of $\mathcal{G}_{i-1}$ and $a_i \notin V_{i-1}$, the hypergraph $\mathcal{G}_i = \mathcal{G}_{i-1} \cup \{C_i \cup \{a_i\}\}$ is a $(k-1)$-hypertree on V_i $(= V_{i-1} \cup \{a_i\})$ and the set of separators of $\mathcal{G}_i$ is $\mathcal{S}_i = \mathcal{S}_{i-1} \cup \{C_i\}$.

From the foregoing it follows that $\mathcal{G}$ $(= \mathcal{G}_{n-k})$ is $(k-1)$-hypertree on V $(= V_{n-k})$. Moreover, $\mathcal{G}$ is finer than $\mathcal{H}$ since every edge of $\mathcal{G}$ is a subset of some edge of $\mathcal{H}$.

It is worth noting that for $\mathcal{G}_1$ we have $\binom{k+1}{2}$ possible choices of (a_1, b_1), and for $\mathcal{G}_i$, $i > 1$, we have at most k choices of C_i. Let $\Gamma(\mathcal{H})$ be the set of outputs of the possible applications of the procedure above with input $\mathcal{H}$. Given a distribution p over V, we want to find a hypergraph $\mathcal{G}^*$ in $\Gamma(\mathcal{H})$ such that $D(p : p_{\mathcal{G}^*})$ is a minimum. Let $\mathcal{G}$ be any hypergraph in $\Gamma(\mathcal{H})$. Since $\mathcal{G}$ is finer than $\mathcal{H}$, by Theorem 2, one has $D(p : p_{\mathcal{G}}) = D(p : p_{\mathcal{H}}) + D(p_{\mathcal{H}} : p_{\mathcal{G}})$ and, since $D(p : p_{\mathcal{H}})$ is fixed, we need to minimize $D(p_{\mathcal{H}} : p_{\mathcal{G}})$. To this end, we now give a useful expression of $D(p_{\mathcal{H}} : p_{\mathcal{G}})$. Let b_i be the unique element of the singleton $B_i - C_i$; thus, $C_i = A_i - \{a_i, b_i\}$ and $\mathcal{G}_i = \mathcal{G}_{i-1} \cup \{A_i - \{b_i\}\}$, $2 \leq i \leq n-k$. We now prove that $D(p_{\mathcal{H}} : p_{\mathcal{G}})$ is equal to the following quantity

$$W = \sum_{1 \leq i \leq n-k} I_p(a_i, b_i | A_i - \{a_i, b_i\})$$

which we call the *weight* of $\mathcal{G}$.

Theorem 4. Let $\mathcal{G}$ be a hypergraph in $\Gamma(\mathcal{H})$. The quantity $D(p_{\mathcal{H}} : p_{\mathcal{G}})$ equals the weight of $\mathcal{G}$.

Proof. First of all, observe that

$$p_{\mathcal{H}} = p_{A_1} \prod_{2 \leq i \leq n-k} \frac{p_{A_i}}{p_{A_i - \{a_i\}}}$$

$$p_{\mathcal{G}} = \frac{p_{A_1 - \{a_1\}} p_{A_1 - \{b_1\}}}{p_{A_1 - \{a_1, b_1\}}} \prod_{2 \leq i \leq n-k} \frac{p_{A_i - \{b_i\}}}{p_{A_i - \{a_i, b_i\}}}.$$

Therefore, the entropy of $p_{\mathcal{H}}$ amounts to

$$E_p(A_1) + \sum_{2 \leq i \leq n-k} E_p(A_i) - E_p(A_i - \{a_i\})$$

and the entropy of $p_{\mathcal{G}}$ amounts to

$$E_p(A_1 - \{a_1\}) + E_p(A_1 - \{b_1\}) - E_p(A_1 - \{a_1, b_1\}) + \sum_{2 \leq i \leq n-k} E_p(A_i - \{b_i\}) - E_p(A_i - \{a_i, b_i\})$$

from which it easily follows that

$$E_{p_{\mathcal{G}}}(V) - E_{p_{\mathcal{H}}}(V) = W.$$

Finally, by Theorem 2 and eq. (2), one has $D(p_{\mathcal{H}} : p_{\mathcal{G}}) = W$. □

By Theorem 4, a simple heuristic to minimize $D(p_{\mathcal{H}} : p_{\mathcal{G}})$ consists in minimizing each term in the additive expression of W. Note that, if $k = n-1$, then $\mathcal{H}$ is the trivial hypergraph on V and a minimum-weight hypergraph $\mathcal{G}$ is an optimal $(n-2)$-hypertree.

We are now in a position to state an algorithm (called `Decompose-and-Reduce`) which, given a k-hypertree $\mathcal{H}$ constructs a $(k-1)$-hypertree $\mathcal{G}$ in $\Gamma(\mathcal{H})$. Instead of starting from a running-intersection ordering of $\mathcal{H}$, the algorithm makes use of a rooted edge-separator tree T_R of $\mathcal{H}$ which is traversed in a top-down way, that is, we visit a node only after visiting its parent. Moreover, the algorithm makes use of the following data structures:

- An edge queue $\mathcal{G}$, which initially is empty.
- An array *state* which is indexed by the set of couples of adjacent vertices of $\mathcal{H}$. For such a couple (u, v), *state*(u, v) = `adjacent` if u and v are adjacent in the hypergraph represented by the current value of $\mathcal{G}$, and *state*(u, v) = `nonadjacent` otherwise. Initially, *state*(u, v) = `nonadjacent` for each couple (u, v).

- For each edge-node A of T_R, a file $F(A)$ of records $[couple, score]$, where *couple* is a couple of elements (u, v) of A and *score* is the conditional mutual information of u and v given $A - \{u, v\}$. The records of $F(A)$ are ordered by non-decreasing values of *score*.
- For each separator-node B of T_R, a list $C(B)$ of couples of distinct elements of B arranged in any order, and a list $L(B)$, which will contain the elements w of B such that $B - \{w\}$ is a clique of the current value of $\mathcal{G}$. Initially, $L(B) := B$ for each separator-node B of T_R.

```
Decompose-and-Reduce
```

STEP 1 (The root of T_R is visited.)
Set $W := 0$.
Take the first record of $F(R)$. Let it be $[(a, b), s]$.
Add $R - \{a\}$ and $R - \{b\}$ to $\mathcal{G}$ and set $W := W + s$.
For each record $[couple, score]$ of $F(R)$ with $couple \neq (a, b)$, set $state(couple) :=$ `adjacent`.

STEP 2 (The other nodes of T_R are visited during a top-down traversal of T_R.)
Distinguish two cases depending on whether the node to visit is a separator-node or an edge node.

> Case 1: the node is a separator-node, say B.
> If $|B| > 2$ then delete from $L(B)$ each element b such that there exists a couple (u, v) in $C(B)$ with $u \neq b$ and $v \neq b$ for which $state(u, v) =$ `nonadjacent`.
>
> Case 2: the node is an edge-node, say A.
> Let B be the parent of A, and let a be the unique element of $A - B$.
> Scan $F(A)$ to find the first of the records $[couple, score]$ where *couple* is of the type (a, w) with $w \in L(B)$. Let it be $[(a, b), s]$.
> Add $A - \{b\}$ to $\mathcal{G}$ and set $W := W + s$.
> For each element w of $B - \{b\}$, set $state(a, w) :=$ `adjacent`.

Remark 2. If we apply the `Decompose-and-Reduce` algorithm with input the edge-separator tree of the trivial hypergraph $\mathcal{H} = \{V\}$, then the output $\mathcal{G}$ is an optimal $(n-2)$-hypertree for p (since $D(p : p_{\mathcal{H}}) = 0$ and $D(p : p_{\mathcal{G}}) = W$).

Example 2. Consider the 3-hypertree $\mathcal{H} = \{abde, acde\}$. The edge-separator tree of $\mathcal{H}$ is shown in Figure 2.

The array *state* has nine components corresponding to the couples (a, b), (a, c), (a, d), (a, e), (b, d), (b, e), (c, d), (c, e) and (d, e), and initially the state of each of them is `nonadjacent`. Assume that the files $F(abde)$ and $F(acde)$ contain the following records:

$F(abde)$

$[(a, d), 0.00934]$	$[(d, e), 0.01372]$	$[(a, b), 0.01581]$
$[(b, e), 0.01829]$	$[(a, e), 0.02611]$	$[(b, d), 0.03553]$

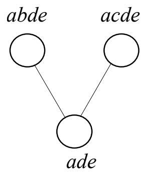

Fig. 2. The edge-separator tree of the 3-hypertree $\{abde, acde\}$.

$F(acde)$

$[(a,c), 0.00540]$	$[(a,d), 0.00676]$	$[(c,e), 0.01298]$
$[(d,e), 0.01303]$	$[(a,e), 0.02942]$	$[(c,d), 0.04887]$

The list $C(ade)$ contains the three couples (a,d), (a,e) and (d,e) and the list $L(ade)$ contains the three elements a, d and e. We apply the `Decompose-and-Reduce` algorithm first with input T_{abde} and, then, with input T_{acde}.

(`Decompose-and-Reduce` with input T_{abde})

STEP 1. After reading the first record $[(a,d), 0.00934]$ of $F(abde)$, we set $\mathcal{G} := (abe, bde)$ and $W := 0.00934$. Moreover, we set

$$state(d,e) := state(a,b) := state(b,e) := state(a,e) := state(b,d) := \texttt{adjacent}.$$

STEP 2. We visit the separator-node ade which is the unique child of the root $abde$. The list $C(ade)$ contains the three couples (a,d), (a,e) and (d,e). Since $state(a,d) =$ `nonadjacent` and $state(a,e) = state(d,e) =$ `adjacent`, we delete e from $L(ade)$ which becomes $L(ade) = ad$. At this point, we visit the edge-node $acde$ which is the unique child of the separator-node ade. The unique element of $acde - ade$ is c. The first record $[couple, score]$ of $F(acde)$ where $couple$ is of the type (c,w) with $w \in L(ade)$ is the record $[(a,c), 0.00540]$. Then, we add the edge $acde - a = cde$ to $\mathcal{G}$, and add 0.00540 to W. Finally, we set $state(c,d) := state(c,e) :=$ `adjacent`. The output of the algorithm is $\mathcal{G} = (abe, bde, cde)$ and $W = 0.01474$.

(`Decompose-and-Reduce` with input T_{acde})

STEP 1. After reading the first record $[(a,c), 0.00540]$ of $F(acde)$, we set $\mathcal{G} := (ade, cde)$ and $W := 0.00540$. Moreover, we set

$$state(a,d) := state(c,e) := state(d,e) := state(a,e) := state(c,d) := \texttt{adjacent}.$$

STEP 2. We visit the separator-node ade which is the unique child of the root $acde$. The list $C(ade)$ contains the three couples (a,d), (a,e) and (d,e). Since $state(a,d) = state(a,e) = state(d,e) =$ `adjacent`, we keep $L(ade) := ade$. At this point, we visit the edge-node $abde$ which is the unique child of the separator-node ade. The unique element of $abde - ade$ is b. The first record $[couple, score]$ of $F(abde)$ where $couple$

is of the type (b, w) with $w \in L(ade)$ is the record $[(a, b), 0.01581$. Then, we add the edge $abde - a = bde$ to $\mathcal{G}$, and add 0.01581 to W. Finally, we set $state(a, b) :=$ `nonadjacent` and $state(b, d) := state(b, e) :=$ `adjacent`. The output of the algorithm is $\mathcal{G} = (ade, cde, bde)$ and $W = 0.02121$.

Remark 3. Example 2 shows that a separator of the input $\mathcal{H}$ need not be an edge of the output $\mathcal{G}$. Recently, Kovács and Szántai [24] have given an algorithm (namely, Algorithm 1) to construct a $(k - 1)$-hypertree from a k-hypertree. Their algorithm is less general then the `Decompose-and-Reduce` algorithm, since it does not apply to the case $k = n - 1$ and, moreover, each separator of the input is required to be an edge of the output.

Finally, we give a procedure for finding a $(k-1)$-hypertree $\mathcal{G}^*$ in $\Gamma(\mathcal{H})$ whose weight, denoted by W^*, is expected to be 'small' (hopefully, a minimum). To achieve this, we repeat the `Decompose-and-Reduce` algorithm for each of the $n-k$ rooted edge-separator trees of $\mathcal{H}$, and take a minimum-weight hypergraph from among the outputs.

```
Greedy Backward

1. Set $\mathcal{G}^*$ to the empty edge list and $W^* := \infty$.
2. Construct the edge-separator tree $T$ of $\mathcal{H}$.
3. For each edge $A$ of $\mathcal{H}$, do:
    3.1 For each couple $(u, v)$ in the array state, set $state(u, v) :=$ nonadjacent.
    3.2 Construct the rooted edge-separator tree $T_A$ of $\mathcal{H}$.
    3.3 Apply Decompose-and-Reduce with input $T_A$.
    3.4 If $W < W^*$ then set $\mathcal{G}^* := \mathcal{G}$ and $W^* := W$.
```

We first analyze the computational complexity of the `Decompose-and-Reduce` algorithm and, then, the computational complexity of the `Greedy Backward` algorithm.

We begin with the size of the input of the `Decompose-and-Reduce` algorithm.

— The rooted edge-separator tree T_R of $\mathcal{H}$. Since $\mathcal{H}$ is a k-hypertree, the number of edge-nodes of T_R is $n - k$. If s is the number of separator-nodes of T_R, then the number of nodes of T_R is $n - k + s$ and the number of arcs of T_R is $n - k + s - 1$. Since $s < n - k$, the number of arcs of T_R is $O(n - k)$. Moreover, since each node can be represented by a vertex list of length k or $k - 1$, we need $O((n - k)k)$ space for storing T_R.

— The array *state*. Since the number of couples of adjacent vertices of $\mathcal{H}$ is the same as the number of edges of the adjacency graph of $\mathcal{H}$, the array *state* has $\left(n - \frac{k+1}{2}\right) k$ components (see Subsection 3.1) so that it requires $O(nk)$ space.

— The F files. For each edge-node A of T_R, since $|A| = k+1$, $F(A)$ contains exactly $\binom{k+1}{2}$ records. Since T_R has $n-k$ edge-nodes, storing the F files requires $O((n-k)k^2)$ space.

— The C and L lists. For each separator-node B of T_R, since $|B| = k$, $C(B)$ contains exactly $\binom{k}{2}$ couples and $L(B)$ contains at most k elements. Since T_R has at most $n-k-1$ separator-nodes, storing the C and L lists requires $O((n-k)k^2)$ space.

From the foregoing it follows that the size of the input is $O((n-k)k^2)$.

We now analyze the single time complexities of STEP 1 and STEP 2 of the `Decompose-and-Reduce` algorithm.

STEP 1 Creating the two edges of $\mathcal{G}$ requires $O(k)$ time, and updating the array *state* requires $O(k^2)$ time. So, the step requires $O(k^2)$ time.

STEP 2 When a separator-node B is visited, we examine the elements of $L(B)$ and, for each element b of $L(B)$, we scan the $\binom{k}{2}$ couples in $C(B)$ to decide whether or not b must be deleted from $L(B)$. Therefore, processing a single separator-node requires $O(k^3)$ time and processing all the separator-nodes requires $O((n-k)k^3)$ time. When an edge-node A is visited, we need to scan the file $F(A)$. Next, we update the value of $\mathcal{G}$, which requires $O(k)$ time. Finally, we update the array *state*, which requires $O(k)$ time. Therefore, processing a single edge-node requires $O(k^2)$ time and processing all the edge-nodes requires $O((n-k)k^2)$ time. So, the step requires $O((n-k)k^3)$ time.

Since the running time of STEP 2 is dominating, the time complexity of the `Decompose-and-Reduce` algorithm is $O((n-k)k^3)$ time.

As to the `Greedy Backward` algorithm, we need to construct the edge-separator tree of $\mathcal{H}$, which requires $O(n-k)k$ time (see Subsection 2.1), and to apply the `Decompose-and-Reduce` algorithm for each edge of $\mathcal{H}$. Therefore, the complexity of the `Greedy Backward` algorithm is $O((n-k)^2k^3)$.

5. A CASE STUDY

The *Jamaican Lizards* dataset [8] contains information on the structural habitat of *grahami* and *opalinus* lizards from Whitehouse, Jamaica. The data consists of observed counts for perch height ($< 5'$, $\geq 5'$), perch diameter ($\leq 2''$, $> 2''$), insolation (sun, shade), and time-of-day (early, midday, late) categories for both grahami and opalinus lizards. We denote the four habitat variables by a, b, c and d respectively, and the species by e. Let p denote the probability distribution over $abcde$ obtained from the sample data. The following are the values of the entropy for the marginals of p [29].

5-set	$E_p(abcde) = 3.21732$		
4-sets	$E_p(abcd) = 2.73327$	$E_p(abce) = 2.27963$	$E_p(abde) = 2.80514$
	$E_p(acde) = 2.59497$	$E_p(bcde) = 2.62262$	

3-sets	$E_p(abc) = 1.77670$	$E_p(abd) = 2.30811$	$E_p(abe) = 1.81318$
	$E_p(acd) = 2.09263$	$E_p(ace) = 1.64171$	$E_p(ade) = 2.17371$
	$E_p(bcd) = 2.10755$	$E_p(bce) = 1.66783$	$E_p(bde) = 2.19907$
	$E_p(cde) = 1.97849$		
2-sets	$E_p(ab) = 1.30243$	$E_p(ac) = 1.12634$	$E_p(ad) = 1.65839$
	$E_p(ae) = 1.17158$	$E_p(bc) = 1.14281$	$E_p(bd) = 1.67593$
	$E_p(be) = 1.19777$	$E_p(cd) = 1.44673$	$E_p(ce) = 1.01847$
	$E_p(de) = 1.55183$		
1-sets	$E_p(a) = 0.64919$	$E_p(b) = 0.66798$	$E_p(c) = 0.47800$
	$E_p(d) = 1.01126$	$E_p(e) = 0.54621$	

Let $\mathcal{H}^{(k)}$ be the optimal k-hypertree, $0 \le k \le 4$. An exhaustive analysis of all k-hypertrees, for each k, gives the following results [17, 29]:

$$\mathcal{H}^{(4)} = \{abcde\} \qquad D(p : p_{\mathcal{H}^{(3)}}) = 0$$

$$\mathcal{H}^{(3)} = \{abde, acde\} \qquad D(p : p_{\mathcal{H}^{(3)}}) = 0.00908$$

$$\mathcal{H}^{(2)} = \{abe, bde, cde\} \qquad D(p : p_{\mathcal{H}^{(2)}}) = 0.02382$$

$$\mathcal{H}^{(1)} = \{ae, be, cd, ce\} \qquad D(p : p_{\mathcal{H}^{(1)}}) = 0.04681$$

$$\mathcal{H}^{(0)} = \{a, b, c, d, e\} \qquad D(p : p_{\mathcal{H}^{(0)}}) = 0.13532$$

The *Jamaican Lizards* dataset was used to test the forward procedure in [17], the two incremental procedures in [29] and in [40] the backward procedure in [36] with the following results:

$$(k = 3)$$

$\mathcal{H}^{(3)}$ [17, 36, 40]

$\mathcal{H} = \{abce, acde\}$ and $D(p : p_{\mathcal{H}}) = 0.01550$ [29]

$$(k = 2)$$

$\mathcal{H}^{(2)}$ [17]

$\mathcal{H} = \{abe, ace, cde\}$ and $D(p : p_{\mathcal{H}}) = 0.02601$ [29]

$\mathcal{H} = \{abe, ade, cde\}$ and $D(p : p_{\mathcal{H}}) = 0.02465$ [40]

$\mathcal{H} = \{abe, bde, cd\}$ and $D(p : p_{\mathcal{H}}) = 0.03263$ [36]

$$(k = 1)$$

$\mathcal{H}^{(1)}$ [17, 29]

$\mathcal{H} = \{ae, b, cd\}$ and $D(p : p_{\mathcal{H}}) = 0.06897$ [36]

In the next three subsections we show that, for k = 4, 3, 2, the `Greedy Backward` algorithm with input $\mathcal{H}^{(k)}$ yields precisely $\mathcal{H}^{(k-1)}$.

5.1. Input $\mathcal{H}^{(4)}$

The edge-separator tree of $\mathcal{H}^{(4)}$ is one-point tree and the `GreedyBackward` algorithm consists of one application of the `Decompose-and-Reduce` algorithm. The file $F(abcde)$ contains the following ten records:

$[(b,c), 0.00908]$ $[(a,c), 0.01137]$ $[(c,e), 0.01298]$ $[(b,d), 0.01557]$

$[(a,d), 0.01710]$ $[(b,e), 0.01829]$ $[(d,e), 0.01888]$ $[(a,b), 0.02178]$

$[(a,e), 0.03102]$ $[(c,d), 0.05427]$

The output of the `GreedyBackward` algorithm is $\mathcal{G}^* = (abde, acde)$ which precisely corresponds to the hypergraph $\mathcal{H}^{(3)}$.

5.2. Input $\mathcal{H}^{(3)}$

The edge-separator tree T of $\mathcal{H}^{(3)}$ was shown in Figure 2. The two applications of the `Decompose-and-Reduce` algorithm with inputs T_{abde} and T_{acde} were discussed in Example 2. It follows that the output of the `GreedyBackward` algorithm is $\mathcal{G}^* = (abe, bde, cde)$ which precisely corresponds to the hypergraph $\mathcal{H}^{(2)}$.

5.3. Input $\mathcal{H}^{(2)}$

The edge-separator tree T of $\mathcal{H}^{(2)}$ is shown in Figure 3.

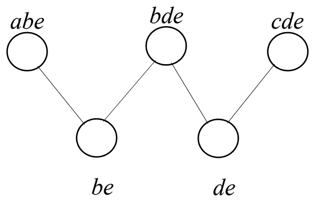

Fig. 3. The edge-separator tree of $\{abe, bde, cde\}$.

The files $F(abe)$, $F(bde)$ and $F(cde)$ contain the following records:

$F(abe)$	$[(a,b), 0.00996]$	$[(b,e), 0.01164]$	$[(a,b), 0.01904]$
$F(bde)$	$[(b,d), 0.00432]$	$[(d,e), 0.00665]$	$[(b,e), 0.01743]$
$F(cde)$	$[(d,e), 0.00871]$	$[(c,e), 0.00881]$	$[(c,d), 0.04560]$

The outputs of the `Decompose-and-Reduce` algorithm with inputs T_{abe}, T_{bde} and T_{cde} are $\mathcal{G} = (ae, be, de, cd)$ with weight 0.02309, $\mathcal{G} = (be, de, ae, cd)$ with weight 0.02309, and $\mathcal{G} = (cd, ce, be, ae)$ with weight 0.02299, respectively. Therefore, the output of the `GreedyBackward` algorithm is $\mathcal{G}^* = (cd, ce, be, ae)$ which precisely corresponds to the hypergraph $\mathcal{H}^{(1)}$.

6. CLOSING NOTE

Suppose that, given a distribution p over a set V of n variables and a real number $\delta > 0$, we want to find an acyclic hypergraph $\mathcal{H}^*$ such that $D(p : p_{\mathcal{H}^*}) < \delta$ and the storage cost of $p_{\mathcal{H}^*}$ is minimum. We can solve this problem as follows. Using the `Decompose-and-Reduce` algorithm with input the trivial hypergraph $\mathcal{H}^{(n-1)} = \{V\}$, we find an optimal $(n-2)$-hypertree $\mathcal{H}^{(n-2)}$ (see Remark 2). Of course, if $D(p : p_{\mathcal{H}^{(n-2)}}) \geq \delta$ then we are done and set $\mathcal{H}^* := \mathcal{H}^{(n-1)}$. If $D(p : p_{\mathcal{H}^{(n-2)}}) < \delta$, then we examine the point partition $\mathcal{H}^{(0)} = \{\{v\} : v \in V\}$. If $D(p : p_{\mathcal{H}^{(0)}}) < \delta$ then we are done and set $\mathcal{H}^* := \mathcal{H}^{(0)}$. Assume that $D(p : p_{\mathcal{H}^{(n-2)}}) < \delta \leq D(p : p_{\mathcal{H}^{(0)}})$. Then, an approximation $\mathcal{H}'$ to $\mathcal{H}^*$ can be obtained in two steps:

(Step 1) Find a k-hypertree $\mathcal{H}$ of minimum tree-width such that $D(p : p_{\mathcal{H}}) < \delta$.

(Step 2) Find an acyclic hypergraph $\mathcal{H}'$ with tree-width k which is a minimal refinement of $\mathcal{H}$ subject to the constraint $D(p : p_{\mathcal{H}'}) < \delta$.

Step 1. We have two alternatives corresponding to the backward and forward approaches.

(*Backward approach*)
Starting with $\mathcal{H}^{(n-2)}$, we repeatedly apply the `GreedyBackward` algorithm until we obtain a $(k-1)$-hypertree $\mathcal{G}^*$ for which $D(p : p_{\mathcal{G}^*}) \geq \delta$. Then, we set $\mathcal{H}$ to the input of the last application of the `GreedyBackward` algorithm.

(*Forward approach*)
We first compute the optimal 1-hypertree $\mathcal{H}^{(1)}$ (that is, the Chow–Liu tree). If $D(p : p_{\mathcal{H}^{(1)}}) < \delta$ then we are done and set $\mathcal{H} := \mathcal{H}^{(1)}$; otherwise, we apply the forward algorithm in [17] until we obtain a k-hypertree $\mathcal{H}$ for which $D(p : p_{\mathcal{H}}) < \delta$.

Step 2. We apply the `GreedyBackward` algorithm with input the edge-separator tree of $\mathcal{H}$, but at each step of the `Decompose-and-Reduce` algorithm, we decompose or reduce an edge of $\mathcal{H}$ only if $D(p : p_{\mathcal{G}}) + W < \delta$. Then, the output $\mathcal{G}^*$ of the `GreedyBackward` algorithm is such that $D(p : p_{\mathcal{G}^*}) + W < \delta$. Needless to say, we can repeat this procedure while maintaining the tree-width equal to k so that ultimately we obtain an acyclic hypergraph $\mathcal{H}'$ with tree-width k which is a minimal refinement of $\mathcal{H}$ and for which $D(p : p_{\mathcal{H}'}) < \delta$.

Suppose to apply the procedure above to the *Jamaican Lizards* dataset with $\delta = 0.01500$. Using any one of the algorithms in [17, 36, 40], at Step 1 we obtain $\mathcal{H}^{(3)} = \{abde, acde\}$ with information divergence 0.00908 and, at Step 2 we obtain the hypergraph $\{abde, cde\}$ with information divergence 0.01448.

(Received August 24, 2011)

REFERENCES

[1] R. Almond and A. Kong: Optimality Issues in Constructing a Markov Tree from Graphical Models. Research Report A-3, Dept. Statistics, Harvard University, 1991.

[2] A. Altmüller and R. M. Haralick: Approximating high dimensional probability disributions. In: Proc. XVII Int. Conf. on Patter Recognitions, 2004.

[3] A. Altmüller and R. M. Haralick: Practical aspects of efficient forward selection in decomposable graphical models. In: Proc. XVI IEEE Int. Conf. on Tools with Artificial Intelligence, 2004, pp. 710–715.

[4] F. R. Bach and M. I. Jordan: Thin junction trees. Adv. in Neural Inform. Proces. Systems *14* (2002), 569–572.

[5] J.-H. Badsberg and F. M. Malvestuto: An implementation of the iterative proportional fitting procedure by propagation trees. Comput. Statist. Data Anal. *37* (2001), 297–322.

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

[7] L. W. Beineke and R. E. Pippert: The enumeration of labelled 2-trees. J. Combin. Theory *6* (1969), 200–205.

[8] Y. Bishop, S. E. Fienberg, and P. W. Holland: Discrete Multivariate Analysis: Theory and Practice. MIT Press, Cambridge 1975.

[9] D. T. Brown: A note on approximations to discrete probability distributions. Inform. and Control *2* (1959), 386–392.

[10] D. Chickering: Learning Bayesian networks is NP-complete. In: Learning from Data, Lecture Notes in Statist. *112* (1996), 121–130.

[11] C. K. Chow and C. N. Liu: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inform. Theory *14* (1968), 462–467.

[12] T. M. Cover: Elements of Information Theory. John Wiley and Sons, 1991.

[13] I. Csiszár and J. Körner: Information Theory. Academic Press, 1981.

[14] P. Dagum and M. Luby: Approximating probabilistic inference in belief networks is NP-hard. Artificial Intelligence *60* (1993), 141–153.

[15] S. Dasgupta: Learning polytrees. In: Proc. XV Conference on Uncertainty in Artificial Intelligence, 1999, pp. 134–141.

[16] A. Deshpande, M. Garofalakis, and M. I. Jordan: Efficient stepwise selection in decomposable models. In: Proc. XVII Conf. on Uncertainty in Artificial Intelligence, 2001, pp. 128–135.

[17] G. Ding, R. F. Lax, J. Chen, P. P. Chen, and B. D. Marx: Comparison of greedy strategies for learning Markov networks of treewidth k. In: Proc. Int. Conf. on Machine Learning: Models, Technologies and Applications, 2007, pp. 294–301.

[18] T. Havránek: On model search methods. In: Proc. IX Symp. on Computational Statistics, 1990, pp. 101–108.

[19] T. Havránek: Simple formal systems in data analysis. In: Proc. Conf. on Symbolic-Numeric Data Analysis and Learning, 1991, pp. 373–381.

[20] F. V. Jensen and F. Jensen: Optimal junction trees. In: Proc. X Conf. on Uncertainty in Artificial Intelligence (R. L. de Mantaras and D. Poole, eds.), 1994, pp. 360–366.

[21] D. Karger and N. Srebro: Learning Markov networks: maximum bounded tree-width graphs. In: Proc. XII ACM-SIAM Symp. on Discrete Mathematics, 2001, pp. 392–401.

[22] T. Kloks: Tree-width. LNCS 842, Springer Verlag, Berlin 1994.

[23] T. Kocka: New algorithm for learning decomposable models. Unpublished manuscript, 2000.

[24] E. Kovács and T. Szántai: Vine copulas as a mean for the construction of high dimensional probability distribution associated to a Markov network. arXiv:1105.1697v1, 2011.

[25] H. H. Ku and S. Kullback: Approximating discrete probability distributions. IEEE Trans. Inform. Theory *15* (1969), 444–447.

[26] S. L. Lauritzen: Graphical Models. Clarendon Press, Oxford 1996.

[27] P. M. Lewis II: Approximating probability distributions to reduce storage requirements. Inform. and Control *2* (1959), 214–225.

[28] F. M. Malvestuto Operations research in the design of statistical databases (in Italian). In: Proc. AIRO Meeting on Operations Research and Computer Science, 1986, pp. 117–130.

[29] F. M. Malvestuto: Approximating discrete probability distributions with decomposable models. IEEE Trans. Systems, Man and Cybernetics *21* (1991), 1287–1294.

[30] F. M. Malvestuto: An axiomatization of loglinear models with an application to the model search. In: Learning from Data, LNS *112* (1996), pp. 175–184.

[31] F. M. Malvestuto: Designing a probabilistic database from a given set of full conditional independences. In: Proc. Workshop on Conditional Independence Structures and Graphical Models, 1999.

[32] F. M. Malvestuto: A hypergraph-theoretic analysis of collapsibility and decomposability for extended log-linear models. Statist. Comput. *11* (2001), 155–169.

[33] F. M. Malvestuto: From conditional independences to factorization constraints with discrete random variables. Ann. Math. and Artificial Intelligence *35* (2002), 253–285.

[34] F. M. Malvestuto: Tree and local computations in a cross-entropy minimization problem with marginal constraints. Kybernetika *46* (2010), 621–654.

[35] C. Meek: Finding a path is harder than finding a tree. J. Artificial Intelligence Res. *15* (2001), 383–389.

[36] M. Mezzini and M. Moscarini: Simple algorithms for minimal triangulation of a graph and backward selection of a decomposable Markov network. Theoret. Comput. Sci. *411* (2010), 958–966.

[37] K. Nunez, J. Chen, P. Chen, G. Ding, R. F. Lax, and B. Marx: Empirical comparison of greedy strategies for learning Markov networks of treewidth k. In: Proc. VII Int. Conf. on Machine Learning and Applications, 2008, pp. 106–113.

[38] J. D. Rose: On simple characterizations of k-trees. Discrete Math. *7* (1974), 317–322.

[39] T. Szántai and E. Kovács: Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. In: Proc. Conf. on Applied Mathematical Programming and Modelling, 2008; also in Ann. Oper. Res. *193* (2012), 71–90.

[40] T. Szántai and E. Kovács: Discovering a junction tree behind a Markov network by a greedy algorithm. arXiv:1104.2762v3, 2011.

[41] R. E. Tarjan and M. Yannakakis: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce hypergraphs. SIAM J. Comput. *13* (1984), 566–579.

[42] N. Wermuth: Analogies between multiplicative models in contingency tables and covariance selection. Biometrics *32* (1976), 95–108.

[43] N. Wermuth: Model search among multiplicative models. Biometrics *32* (1976), 253–256.

[44] Y. Xiang, S. K. M. Wong, and N. Cercone: A "microscopic" study of minimum entropy search in learning decomposable Markov networks. Mach. Learning *26* (1997), 65–72.

Francesco M. Malvestuto, Department of Informatics, Faculty of Information Engineering, Informatics and Statistics, Sapienza University of Rome, Via Salaria 113, 00198 Rome. Italy.
e-mail: malvestuto@di.uniroma1.it