Using imsets for learning Bayesian networks*

Jiří Vomlel [2,1]

[1]Faculty of Management
University of Economics
Jindřichův Hradec
Czech Republic
http://www.utia.cz/vomlel

Milan Studený [2]

[2]Inst. of Inform. Th. and Automation
Academy of Sciences
Prague
Czech Republic
http://www.utia.cz/studeny

Abstract

This paper describes a modification of the greedy equivalence search (GES) algorithm [1]. The presented modification is based on the algebraic approach to learning suggested in [8]. The states of the search space are standard imsets. Each standard imset represents an equivalence class of Bayesian networks. For a given quality criterion the database is represented by the respective data imset. This allows a very simple update of a given quality criterion since the moves between states are represented by differential imsets. We exploit a direct characterization of lower and upper inclusion neighborhood [6], which allows an efficient search for the best structure in the inclusion neighborhood. The algorithm was implemented in R and is freely available [3].

1 Bayesian networks

One of the fundamental tasks in the area of probabilistic modeling in artificial intelligence is *learning models from data*. If the regarded class of probabilistic models is the class of *Bayesian networks* then learning is a bit tricky since two different Bayesian networks may represent the same joint probability distribution. Before we get into details let us introduce the basic notion.

Definition 1 A *directed graph* is a pair $G = (N, \mathcal{E})$, where N is the set of nodes of G and $\mathcal{E} \subseteq (a, b) \in N \times N : a \neq b$ is the set of directed edges of G. $a \to b = (a, b)$ will denote the directed edge from a to b.

*Both authors were supported by the Ministry of Education of the Czech Republic through grant nr. 1M0572 DAR. The first author was also supported through grant nr. 2C06019 Zimolez.

Definition 2 A *directed cycle* in a graph G is a sequence $c_1, \ldots, c_n, c_{n+1} \equiv c_1, n \geq 3$ of nodes in G such that $c_1, \ldots, c_n$ are distinct and for $i = 1, \ldots, n-1$: $(c_i, c_{i+1}) \in \mathcal{E}$.

The structural part of a *Bayesian network* [5, 4] is defined by an *acyclic directed graph* (DAG) $G = (N, \mathcal{E})$, i.e., a directed graph without any directed cycle. The graph encodes conditional independence relations between variables $X_i, i \in N$ corresponding to graph nodes.

Definition 3 Two nodes a and b in a DAG G are *d-separated* by a set C if for all paths between a and b there is a node c ($c \neq a$ and $c \neq b$) such that either:

- the path contains a node $c \in C$, in which edges do not meet "head-to-head" or
- the path contains a node c in which edges "head-to-head" and neither c nor any of its descendants belong to C.

In this paper we will regard only discrete variables, i.e., each variable X_i will take its values x_i from a finite set $\mathbb{X}_i$. For each node $i \in N$ its set of parents in graph G is defined as the set of nodes from which there is a directed edge to i in G, i.e., $pa_G(i) = \{j : (j \to i) \in \mathcal{E}\}$. A conditional probability table $P_{i|pa_G(i)}$ is defined for each variable $X_i, i \in N$. Let for $A \subseteq N$ symbol x_A denote a vector of values of multidimensional variable $X_A = (X_i)_{i \in A}$. Further let for a particular vector of values $(x_i, x_{pa_G(i)})$ the symbol $p(x_i \mid x_{pa_G(i)})$ denote the conditional probability of $X_i = x_i$ given $X_{pa_G(i)} = x_{pa_G(i)}$ from the conditional probability table $P_{i|pa_G(i)}$.

The joint probability represented by a Bayesian network is defined for all x_N by the product formula

$$p(x_N) \quad = \quad \prod_{i \in N} p(x_i \mid x_{pa_G(i)}) \ ,$$

where $x_i, x_{pa_G(i)}$ for $i \in N$ are corresponding values of vector x_N.

Definition 4 Let P be a discrete probability distribution over X_N and A, B, C be pairwise disjoint subsets of N. The *conditional independence (CI) statement* $A \perp\!\!\!\perp B \mid C$ is induced by probability distribution P if for all x_A, x_B, x_C such that $P(x_C) > 0$

$$P(x_A, x_B \mid x_C) \quad = \quad P(x_A \mid x_C) \cdot P(x_B \mid x_C) \ .$$

Lemma 1 *Let P be the joint probability distribution P represented by a Bayesian network with structure defined by a DAG G. If nodes a and b are d-separated by a set C in the DAG G then $a \perp\!\!\!\perp b \mid C$ is induced by P.*

2 Essential graphs

Note that two Bayesian networks with different DAGs may represent the same set of CI-statements. We say that Bayesian networks with DAGs representing the same set of CI-statements belong to an *equivalence class*. To characterize DAGs representing the same set of CI-statements we will use the graph notion of *immorality* and *underlaying graph*.

Definition 5 An *immorality* in a DAG G is a induced subgraph of G for a set $\{a, b, c\}$, where a, b, c are distinct nodes of G such that there are edges $a \to c$ and $b \to c$ and there is no edge between a and b in G.

Definition 6 An *underlaying graph* of a DAG is the undirected graph that has the same set of nodes and all directed edges $a \to b$ are replaced by undirected edges $a - b$.

Lemma 2 *Bayesian networks belong to the same equivalence class iff they have the same underlaying graph and the same set of immoralities.*

Definition 7 The *essential graph* G^* of an equivalence class $\mathcal{G}$ of DAGs over N is a hybrid graph (i.e., a graph with both directed and undirected edges) over N defined as follows:

- $a \to b$ in G^* if $a \to b$ in G for every $G \in \mathcal{G}$,
- $a - b$ (or, equivalently we will write $\{a, b\}$) in G^* if $\exists\, G_1, G_2 \in \mathcal{G}$ such that $a \to b$ in G_1 and $a \leftarrow b$ in G_2.

3 Standard imsets

Let $\mathcal{P}(N)$ denote the power set of N and $\mathbb{N}$ be the set of all natural numbers. Function $m : \mathcal{P}(N) \mapsto \mathbb{N}$ is sometimes called multiset. Imset is an abbreviation from **I**nteger valued **M**ulti**SET** [7].

Definition 8 Let $\mathbb{Z}$ be the set of all integers. *Imset* is a function $u : \mathcal{P}(N) \mapsto \mathbb{Z}$.

We will specify imsets using a special convention. We use Kronecker's symbol δ to denote an indicator function defined for $A, B \subseteq N$ as follows:

$$\delta_A(B) \quad = \quad \begin{cases} 1 & \text{if } A = B, \\ 0 & \text{otherwise.} \end{cases}$$

Then, given an imset u, one has

$$\forall B \subseteq N: \; u(B) \quad = \quad \sum_{A \subseteq N} u(A) \cdot \delta_A(B)$$

which can be abbreviated as

$$u \quad = \quad \sum_{A \subseteq N} c_A \cdot \delta_A$$

where $c_A = u(A) \in \mathbb{Z}$ is a the respective coefficient for every $A \subseteq N$.

Definition 9 Let $C \subseteq N$, $a, b \in N \setminus C$, and $a \neq b$. The *elementary imset* corresponding to $a \perp\!\!\!\perp b \mid C$ is given by the formula

$$u_{\langle a,b|C\rangle} \quad = \quad \delta_{\{a,b\}\cup C} + \delta_C - \delta_{\{a\}\cup C} - \delta_{\{b\}\cup C} \ .$$

An algebraic uniquely determined representative of an equivalence class of Bayesian networks is the *standard imset* [7].

Definition 10 The *standard imset* for a graph G is given by the formula

$$u_G \quad = \quad \delta_N - \delta_\emptyset + \sum_{a\in N} \left\{\delta_{pa_G(a)} - \delta_{\{a\}\cup pa_G(a)}\right\} \ . \tag{1}$$

Two DAGs G and H implies an equivalent set of CI-statements iff $u_G = u_H$. Thus, standard imsets can serve as unique representatives of the respective CI model. Typically, standard imsets, viewed as vectors, have many zero components. Therefore, if we store only the non-zero components then they can be effectively kept in the memory of a computer.

4 Bayesian Information Criterion

In this paper we are interested in learning the structure (DAG) of a Bayesian network from data. A class of methods for learning the structure of Bayesian networks is based on the maximization of a quality criterion. For simplicity, we will deal with the Bayesian Information Criterion (BIC) only.

Let $D = \{x^m, m = 1, \ldots, M\}$ be the learning dataset, where x^m is a vector of values of variable $X = \{X_i\}_{i=1}^N$. Further, let for $A \subseteq N$, $N(A)$ be the distribution defined by the counting function that provides the number $n(X_A = x_A)$ of data vectors for each particular combination x_A of values of variables from set X_A. Note that $\sum_{x_A} n(X_A = x_A) = M$. Therefore $\frac{n(A)}{M}$ is a probability distribution.

Let $r(i)$ denote number of states of variable X_i and $q(i, G)$ denote number of parent configurations for parents $X_{pa(i)}$ of variable X_i. For $i = 1, \ldots, N$, $j = 1, \ldots, r(i)$, and $k = 1, \ldots, q(i, G)$ we will use the abbreviation $n(i, j, k)$ for $n(X_i = x_j, X_{pa(i)} = x_k)$, i.e., for the occurrence of the configuration $(X_i = x_j, X_{pa(i)} = x_k)$ in the learning dataset D. Similarly, we define the abbreviation $n(i, j)$.

The likelihood of dataset D given graph G is the probability of D being generated from the Bayesian network model that has the structure given by the DAG G and defines the joint probability distribution P, i.e.,

$$P(D \mid G) \quad = \quad \prod_{m=1}^{M} P(X = x^m) \ .$$

Lemma 3 (MLL) *The value of maximal log-likelihood (MLL) for a Bayesian network with a DAG G is*

$$MLL(G \mid D) \quad = \quad \sum_{i=1}^{N} \sum_{k=1}^{r(i)} \sum_{j=1}^{q(i,G)} n(i, j, k) \log \frac{n(i, j, k)}{n(i, j)}$$

Let $d(G)$ denote the number of free parameters in the Bayesian network model with graph G. It is given by

$$d(G) \quad = \quad \sum_{i=1}^{N}(r(i)-1)\cdot q(i,G) \ .$$

Now, we are ready to define the Bayesian Information Criterion (BIC).

Definition 11

$$BIC(G \mid D) \quad = \quad MLL(G \mid D) - \frac{\log M}{2} d(G)$$

The BIC can be efficiently computed for each Bayesian network with the structure given by a DAG G as the scalar product $\langle u_G, v\rangle$ of the standard imset u_G and a data imset v plus a constant that depends only on data[1]. The data imset v is defined for each $A \subseteq N$ as

$$v_A \quad = \quad d \cdot H\left(\frac{n(A)}{d}\right) + \frac{\log(M)}{2} \cdot \left(|A| - 1 + \prod_{i\in A} r(i) - \sum_{i\in A} r(i)\right) \quad , \quad (2)$$

where H is the Shannon entropy function

$$H(P(A)) \quad = \quad - \sum_{x_A : p(x_A)>0} p(x_A)\log p(x_A) \ .$$

Note that the value of the data imset v can be computed for a set $A \subseteq N$ only if it is needed. The computed values can be stored in a cache memory so that they are computed only once.

In [7] the data imset was derived in terms of multiinformation, while here we provide it in terms of entropy. In order to derive the formula similar transformations to those in [7, Chapter 8] are performed.

5 Greedy equivalence search in the space of standard imsets

Because the direct maximization of a quality criterion is typically infeasible, methods of local search were developed. The basic idea of the method proposed in [1] is to use the concept of *inclusion neighborhood* for representatives of considered CI-models and search for a local maximum of the criterion with respect to the neighborhood structure.

Definition 12 Let $\mathcal{M}_G$ denote the set of CI-statements generated by a DAG G. Given two DAGs K, L over N, we say that they are *inclusion neighbors* and write $\mathcal{M}_K \sqsubset \mathcal{M}_L$ if $\mathcal{M}_K \subset \mathcal{M}_L$ and there is no DAG G such that $\mathcal{M}_K \sqsubset \mathcal{M}_G \sqsubset \mathcal{M}_L$. We say then that $\mathcal{M}_L$ is an upper neighbor of $\mathcal{M}_K$ or, dually, that $\mathcal{M}_K$ is a lower neighbor of $\mathcal{M}_L$.

[1] Since the constant does not depend on the model we need not compute it when searching for the best model.

Table 1: The main loop

bestModel(Data)
`new` $u = imset(emptyGraph(N))$ // standard imset
`new` $v = dataImset(Data)$ // data imset
Found:=TRUE
`while` (Found)
(Found,u) :=*bestNb*(u, v, "lower")
Found=TRUE
`while` (Found)
(Found,u) :=*bestNb*(u, v, "upper")
`return` *essentialGraph*(u)

The basic idea of the algebraic approach to learning Bayesian networks is to use the standard imset as a unique representative of each equivalence class. The advantage of this approach is that every imset can be interpreted as a vector and the BIC appears to be affine (= shifted linear) functions of the standard imset. In the case of the BIC one has formula (2) for the respective data vector.

Our implementation of the Greedy Equivalence Search (GES) algorithm starts with the imset corresponding to the empty graph. As well as Chickering's implementation of the GES it has two stages: (1) deleting CI-statements and (2) adding CI-statements. See Table 1.

The move between two neighboring models is characterized by a simple elementary imset, which is the difference of respective standard imsets. Therefore, each move can be interpreted in terms of a CI statement $a \perp\!\!\!\perp b \mid C$. In each step of the GES algorithm:

- we select the model from the lower (resp. upper in the second stage) inclusion neighborhood that maximizes the improvement of the criteria,
- if there is no better model than the current one we start the second stage or we terminate if we are in the second stage already.

The respective change in the value of the BIC takes a neat form of the scalar product of two vectors:

$$\langle v, u_{\langle a,b|C\rangle}\rangle \quad = \quad \sum_{A\subseteq N} v(A)\cdot u_{\langle a,b|C\rangle}(A)\,.$$

See Tables 2 and 3 for the actual implementation of the search of a best neighbor.

6 Inclusion neighborhood

In this section we describe an efficient implementation of the inclusion neighborhood. A characterization of inclusion neighborhood in terms of standard imsets

Table 2: Search for the best lower or upper neighbor

bestNb(u, v, which)

Found:=FALSE
G:=*essentialGraph*(u)
$\mathcal{E}$:=*edges*(G)
N:=*nodes*(G)
$t^\star := 0$
`for` $\{a, b\} \in \{\{a, b\} : a \in N, b \in N, a \neq b\}$
  `if` $((a, b) \notin \mathcal{E}) \wedge ((b, a) \notin \mathcal{E}) \wedge (\{a, b\} \notin \mathcal{E})$ `then`
    `if` which="lower" `then`
      $\mathcal{C} :=$ *condSetsLower*(a, b, G)
      $(C, t) :=$ *bestCondSet*$(a, b, \mathcal{C}, v, \text{which})$
    `else`
      `if` which="upper" `then`
        $\mathcal{C} :=$ *condSetsUpper*(a, b, G)
        $(C, t) :=$ *bestCondSet*$(a, b, \mathcal{C}, v, \text{which})$
    `if` $t > t^\star$ `then`
      $a^\star := a$
      $b^\star := b$
      $C^\star := C$
      $t^\star := t$
`if` $(t^\star > 0)$
  Found=TRUE
  $w := \delta_{C^\star} + \delta_{\{a^\star\} \cup \{b^\star\} \cup C^\star} - \delta_{\{a^\star\} \cup C^\star} - \delta_{\{b^\star\} \cup C^\star}$
  `if` (which="lower") `then` $u := u - w$
                       `else` $u := u + w$
`return`(Found,u)

Table 3: Search for the best conditioning set

bestCondSet$(a, b, \mathcal{C}, v, \text{which})$

first:=TRUE
$t^\star := 0$
`for` $C \in \mathcal{C}$
  $t := v_C + v_{\{a\} \cup \{b\} \cup C} - v_{\{a\} \cup C} - v_{\{b\} \cup C}$
  `if` which="lower" `then` $t := -t$
  `if` first $\vee$ $(t > t^\star)$ `then`
    first:=FALSE
    $t^\star := t$
    $C^\star := C$
`return` $(C^\star, t^\star)$

is an open problem. Therefore we used the transformation of imsets to essential graphs [9, 10] and the characterization of inclusion neighborhood in terms of the essential graph [6]. The desired properties of the characterization are that (1) all generated neighbors are valid CI-models (i.e. correspond to a Bayesian network model) and (2) all generated neighbors represent different CI-models (i.e. each CI-models is generated exactly once).

First, we describe the lower neighborhood corresponding to deleting a CI-statement $a \perp\!\!\!\perp b \mid C$, which corresponds to adding an edge between a and b and possibly directing some undirected edges. The explicit formula is provided in Table 4.

Let us explain the graph notion used in Table 4. A set A is complete in an essential graph G if for all $a, b \in A, a \neq b$ there is an undirected edge (a, b) in G. Clique is a maximal complete set (with respect to the set inclusion) in G. Function *cliques*(G) returns the set of all cliques of the graph G. Function $ne_G(a)$returns the set of neighbor nodes of node a, i.e., the set of nodes connected by an undirected edge, in the graph G. We say that a path is *descending* if it consists of edges directed toward the goal or from undirected edges. Similarly, a path is *strictly descending* if it consists of edges directed toward the goal. Function *setOfAncestors*(b, S, G) (or *setOfStrictAncestors*(b, S, G)) returns all nodes for which there exists a descending path (or strictly descending path, respectively) to node b in the essential graph G such that all nodes in the path except the first are outside the set S. G_S is the graph generated from the graph G by the set of edges S. Function *neOfAll*(S, G) returns the set of nodes that are connected by an undirected edge to all nodes from the set S in the essential graph G.

The characterization of upper neighborhood corresponding to adding a CI-statement $a \perp\!\!\!\perp b \mid C$ is much simpler. Note that, in this case, it corresponds to deleting an edge between a and b and possibly directing some undirected edges. The explicit formula is provided in Table 5.

7 Experimental results

We implemented the described algorithm in R - a free software environment for statistical computing and graphics. Our algorithm is also freely available [3]. We performed the following experiment[2] with five different Bayesian network models. We either created a Bayesian network ourselves (abcde and abcdefghi) or used some of Bayesian networks provided with Hugin [2] (asia, boerlage92, and alarm). We used the Hugin [2] tool to generate data samples from these models. In case of networks abcde, asia, and abcdefghi we generated 10^4 data samples. In case of Boerlage92 and alarm we generated 10^6 data samples.

Then we run our implementation of the learning algorithm in R and counted the number of evaluated neighbors and the number of selected neighbors. See Table 6 for the results. For all models except the alarm model there was no

[2]The experiment demonstrate the size of the search space of our implementation. It cannot serve as an evaluation of algorithm's ability to discover the original model.

Table 4: Construction of the list of all conditioning sets C when deleting a CI-statement $a \perp\!\!\!\perp b \mid C$.

condSetsLower(a, b, G)

$R_a := setOfAncestors(b, ne_G(a), G) \cap ne_G(a)$
$R_b := setOfAncestors(a, ne_G(b), G) \cap ne_G(b)$
$A_a = setOfAncestors(a, \emptyset, G)$
$A_b = setOfAncestors(b, \emptyset, G)$
$A'_a = setOfStrictAncestors(a, \emptyset, G)$
$A'_b = setOfStrictAncestors(b, \emptyset, G)$
$M_a := (ne_G(a) \setminus R_a) \cap neOfAll(R_a, G)$
$M_b := (ne_G(b) \setminus R_b) \cap neOfAll(R_b, G)$
`if` $a \in A'_b$ `then`
  $C_b := \emptyset, \mathcal{S}_b := \emptyset$
  `if` $isComplete(G_{R_b})$ `then` $C_a := pa_G(b) \cup R_b, \mathcal{S}_a := cliques(G_{M_b})$
    `else` $C_a := \emptyset, \mathcal{S}_a := \emptyset$
`else`
  `if` $b \in A'_a$ `then`
    $C_a := \emptyset, \mathcal{S}_a := \emptyset$
    `if` $isComplete(G_{R_a})$ `then` $C_b := pa_G(a) \cup R_a, \mathcal{S}_b := cliques(G_{M_a})$
      `else` $C_b := \emptyset, \mathcal{S}_b := \emptyset$
  `else`
    `if` $a \in A_b$ `then`
      `if` $b \in A_a$ `then`
        $C_a := pa_G(b) \cup R_b, \mathcal{S}_a := cliques(G_{M_b})$
        $C_b := pa_G(a) \cup R_a, \mathcal{S}_b := cliques(G_{M_a})$
      `else`
        $C_a := pa_G(b), \mathcal{S}_a := \mathcal{C}_b$
        `if` $isComplete(G_{R_a})$ `then` $C_b := pa_G(a) \cup R_a, \mathcal{S}_b := cliques(G_{M_a})$
          `else` $C_b := \emptyset, \mathcal{S}_b := \emptyset$
    `else`
      `if` $b \in A_a$ `then`
        $C_b := pa_G(a), \mathcal{S}_b := \mathcal{C}_a$
        `if` $isComplete(G_{R_b})$ `then` $C_a := pa_G(b) \cup R_b, \mathcal{S}_a := cliques(G_{M_b})$
          `else` $C_a := \emptyset, \mathcal{S}_a = \emptyset$
      `else`
        $C_a := pa_G(b), \mathcal{S}_a := cliques(G_{M_b})$
        $C_b := pa_G(a), \mathcal{S}_b := cliques(G_{M_a})$
$\mathcal{C}_a := \{C_a \cup B : \exists D \in \mathcal{S}_a, B \subseteq D\}$
$\mathcal{C}_b := \{C_b \cup B : \exists D \in \mathcal{S}_b, B \subseteq D\}$
$\mathcal{C} := \mathcal{C}_a \cup \mathcal{C}_b$
`return` $\mathcal{C}$

Table 5: Construction of the list of all conditioning sets C when adding a CI-statement $a \perp\!\!\!\perp b \mid C$.

condSetsUpper(a, b, G)
`if` $(a,b) \in \mathcal{E}$ `then`
$C := pa_G(b) \setminus \{a\}$
$M := ne_G(b)$
`else`
`if` $(b,a) \in \mathcal{E}$ `then`
$C := pa_G(a) \setminus \{b\}$
$M := ne_G(a)$
`else` // $\{a,b\} \in \mathcal{E}$
$C := pa_G(a) \setminus \{b\}$
$M := ne_G(a) \cap ne_G(b)$
$\mathcal{S} :=$ *cliques*(G_M)
$\mathcal{C} := \{C \cup B : \exists D \in \mathcal{S}, B \subseteq D\}$
`return` $\mathcal{C}$

improvement after the first stage. Therefore, the number of selected upper neighbors is zero.

In case of the abcde, asia, and abcdefghi models the algorithm found the same CI-model as the original one. In case of boerlage92 one edge was missing and few were undirected instead of directed or reversely directed. But when looking back in the conditional probability tables of the original model we realized that the corresponding dependencies were very weak, so the learned model can be regarded as a correct one. The original alarm network has 46 edges. The learned network did not miss any edge, but 4 edges were reversed[3] and 5 edges were added. It means that in total it had 51 edges.

8 Conclusions

In this paper we have described an efficient implementation of the greedy equivalence search algorithm. The basic idea is to use the standard imset [7] as a unique representative of each equivalence class. The search in the inclusion neighborhood is simple since the change in a criteria value is just a scalar product of the elementary imset corresponding to the move and the data imset. Since we use the efficient characterization of the inclusion neighborhood from [6] all generated models correspond to a Bayesian network model and for each equivalence class from the inclusion neighborhood exactly one model is generated and evaluated. We implemented the algorithm in R. Our implementation is freely available [3].

[3] Of course, we compare the essential graphs.

Table 6: The numbers of evaluated neighbors (column 4) and the number of selected neighbors (column 3) for five models with different number of variables (column 2). For each model in the first row the numbers are for the lower neighbors and in the second row for the upper neighbors.

Model	(2)	(3)	(4)
abcde	5	4	60
		0	4
asia	8	8	389
		0	8
abcedefghi	9	10	621
		0	10
boerlage92	23	35	19607
		0	47
alarm	37	52	65787
		1	128

In future we would like to characterize better the search space. It would be very useful to confirm the conjecture that every standard imset is an extreme point of the polytope consisting of convex combinations of standard imsets. If this is true then the methods of linear programming can possibly be applied to the problem of learning Bayesian networks.

References

[1] D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[2] Hugin Expert A/S. `http://www.hugin.com`.

[3] `imset.R` - a suite of functions for R. Available at `http://staff.utia.cas.cz/vomlel/imset/`.

[4] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Verlag, 2001.

[5] J. Pearl. *Probabilistic Reasoning in Intelligent Systems - Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[6] M. Studený. Characterization of inclusion neighbourhood in terms of the essential graphs. *International Journal of Approximate Reasoning*, 38(3):283–309, 2005.

[7] M. Studený. *On Probabilistic Conditional Independence Structures*. Springer London, 2005.

[8] M. Studený. An algebraic approach to structural learning Bayesian networks. In B. Bouchon-Meunier and R. R. Yager, editors, *Proceedings of the 11th International Conference IPMU 2006*, pages 2284–2291, Paris, France, 2006. Editions EDK.

[9] M. Studený and J. Vomlel. Transition between graphical and algebraic representatives of Bayesian network models. In *Proceeding of the 2nd European Workshop on Probabilistic Graphical Models (PGM'04)*, pages 193–200, Leiden, the Netherlands, 2004.

[10] J. Vomlel and M. Studený. *Advances in Bayesian Networks*, chapter Graphical and algebraic representatives of conditional independence models. Springer-Verlag, 2007.