INSTITUTO TECNOLOGICO DE BUENOS AIRES
UNIVERSIDAD PRIVADA

LEARNING BAYESIAN NETWORKS SKELETON: A COMPARISON BETWEEN TPDA AND PMMS ALGORITHM

- TOMÁS GROPPO PARISI -

TUTOR:

- PR. ALEXANDRE AUSSEM, UCBL (UNIVERSITE CLAUDE BERNARD LYON 1)

CO-TUTOR:

- PHD. RAMÓN GARCÍA MARTINEZ, ITBA (INSTITUTO TECNOLÓGICO DE BUENOS AIRES)

- 2006 -

TABLE OF CONTENTS

ABSTRACT

Learning the bayesian network structure from a database is an NP-Hard problem for which the existent learning algorithms generally have exponential complexity. During this work in the Master, I did a bibliographic research as well as a comparison between two recent algorithms called TPDA and PMMS (2005) that learns the skeleton of bayesian networks from data. These algorithms have the advantage of having polynomial complexity, and provide good results for learning. After having done a theoretical analysis of the algorithms, I continue with an empiric analysis that consisted in testing these algorithms on data generated from networks knew by the scientific community (I used ASIA and ALARM networks). These tests have been made with the help of the toolboxes developed in Matlab (FullBNT, BNT – SLP and CausalExplorer). The results I have gotten by this analysis have permitted me to make some interesting conclusions about the efficiency and the limits of application of these algorithms.

Key Words: Statistical learning, Data-mining, Knowledge extraction.

RESUME

L'apprentissage du squelette d'une réseau bayésien à partir d'une base de données est un problème NP-Difficile pour lequel les algorithmes d'apprentissage existant sont généralement de complexité exponentielle. Au cours de ce stage du Master, j'ai effectué une recherche bibliographique ainsi qu'une comparaison entre deux algorithmes récents d'apprentissage du skeleton des réseaux bayésiens qui sont TPDA et PMMS (2005). Ces algorithmes on l'avantage d'avoir une complexité polynomiale, et fournissent de bons résultats d'apprentissage. Apres avoir effectué une analyse théorique des algorithmes, j'ai poursuivi sur une analyse empirique qui consisté à mettre en pratique ces algorithmes sur des données générées à partir de réseaux connus dans la communauté scientifique (J'ai utilisé le réseaux ASIA et ALARM). Cela a été fait a l'aide de boite à outils développé en Matlab (FullBNT, SNT – SLP et CausalExplorer). Les résultats obtenus par cette analyse m'ont permit de tirer des conclusions intéressantes quant à l'efficacité et les limites d'application de ces algorithmes.

Mots clef : Apprentissage statistique, Data-mining, Extraction des connaissances.

RESUMEN

El aprendizaje del esqueleto de una red bayesiana a partir de una base de datos es un problema NP-Dificil para el cual los algoritmos de aprendizaje existentes son generalmente de complejidad exponencial. En el transcurso de la pasantía del Master, he efectuado una investigación bibliográfica así como una comparación entre dos algoritmos recientes de aprendizaje del esqueleto de las redes bayesianas. Ellos son TPDA y PMMS (2005). Estos algoritmos tienen la ventaja de tener una complejidad polinomial y proveen buenos resultados de aprendizaje. Luego de haber efectuado un análisis teórico de los algoritmos, he continuado con un análisis empírico. Este consistió en testear estos algoritmos sobre datos generados a partir de redes conocidas por la comunidad científica (he utilizado la red ASIA y ALARM). Esto fue realizado con la ayuda de las cajas de herramientas desarrolladas en Matlab (FullBNT , BNT – SLP y CausalExplorer). Los resultados obtenidos por este análisis me permitieron formular conclusiones interesantes en cuanto a la eficacia y a los límites de aplicación de estos algoritmos.

Palabras clave: Aprendizaje estadístico, Data-mining, Extracción de conocimientos.

1. INTRODUCTION

This project is the result of several months learning and working with bayesian networks. I started with some basic concepts of them which made me understand the magnitude of this tool. I was shocked by the way they arrived to show all the information of complex databases in a simple and intuitive graph. I soon realize it was not so easy to transform the databases into bayesian networks. This problem is NP-Hard[1] and several researchers had tried to solve it with different combinational optimization methods. The complex of the problem leads to complex algorithms which generally needs exponential complexity in the worst case. However, TPDA (2002) and PMMS (2005) are two recent algorithms that had obtained polynomial complexity. They are polynomial variants of the two most recognize algorithms for learning bayesian networks which are PC (1993) and MMHC (2005). The publication of these algorithms in important revues is a proof of this recognizance and the recent publication of MMHC (Max Min Hill Climbing) in "Machine Learning" revue this year shows the current importance of them.

When I was analysing these polynomials algorithms I realized that the only comparison between TPDA and PMMS was the one made by the authors of PMMS. In that article, they claimed that PMMS would work better than TPDA when the amount of data was small; consequently, in their tests they stopped just when TPDA started to outperform PMMS (with 5.000 data entries). To verify their statement, I decided to test these algorithms with a big amount of data (until 20.000 data entries). For TPDA algorithm I found some preliminary implementations in two Matlab toolboxes, but for PMMS there was no one. Making some research I took knowledge of a preliminary development of PRISMa laboratory for PMMS algorithm also in Matlab code. After learning how to use Matlab and all the toolboxes for bayesian networks I was finally in condition to test the two algorithms. I chose the ASIA and ALARM network to test them because they are the most used benchmark networks in the area and many researchers have used them to evaluate their algorithms. I made 8 sub-databases with different number of data for each network and I tested the power of both algorithms to learn the skeleton of these networks. The results finally show that PMMS generally outperforms TPDA even with big amounts of data.

This paper is organized in 6 sections. In the following section I will define Bayesian Networks and I will give some relevant information about them. In the third section I will examine how TPDA and PMMS algorithms work; and in section four I will give the results of these algorithms when learning the skeleton of ASIA and ALARM networks. Finally, I will present some conclusions which are based on my research.

[1] A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem (Nondeterministic Polynomial time problem). NP-hard therefore means "at least as hard as any NP-problem," although it might be harder.

2. PRELIMINARIES

2.1 BAYESIAN NETWORK

Nowadays acquiring information is relatively easy, and the new challenge is to transform this information into knowledge. We can find different techniques for this purpose although this paper treats solely the method associated with Learning Bayesian Networks (for a comparison between the different techniques see annexe A).

The Bayesian network is a powerful knowledge representation and reasoning tool introduced by [Kim & Pearl, 1987], [Lauritzen & Speigelhalter, 1988], [Jensen, 1996] and [Jordan, 1998]. The formal definition of this network is:

Definition 1*. B=(G, P) is a* ***bayesian network*** *if G=(X,E) is a directed acyclic graph (DAG) where the set of nodes represents a set of random variables* $X = \{X_1, \ldots ,X_n\}$*, and if* $P_i=[P(X_i/X_{Pa(Xi)})]$ *is the matrix containing the conditional probability of node i given the state of its parents* $Pa(X_i)$*.*

2.2 LEARNING BAYESIAN NETWORKS

The problem of learning the most probable a posteriori Bayesian network (BN) from data under certain broad conditions is worst-case NP-hard (Chickering, Meek, & Heckerman 2003). This problem involves two subtasks:

- Learning the structure of the network: determining the dependant elements
- Learning the parameters: the strength of these dependencies, as encoded by the entries in the condition of probabilities between variables.

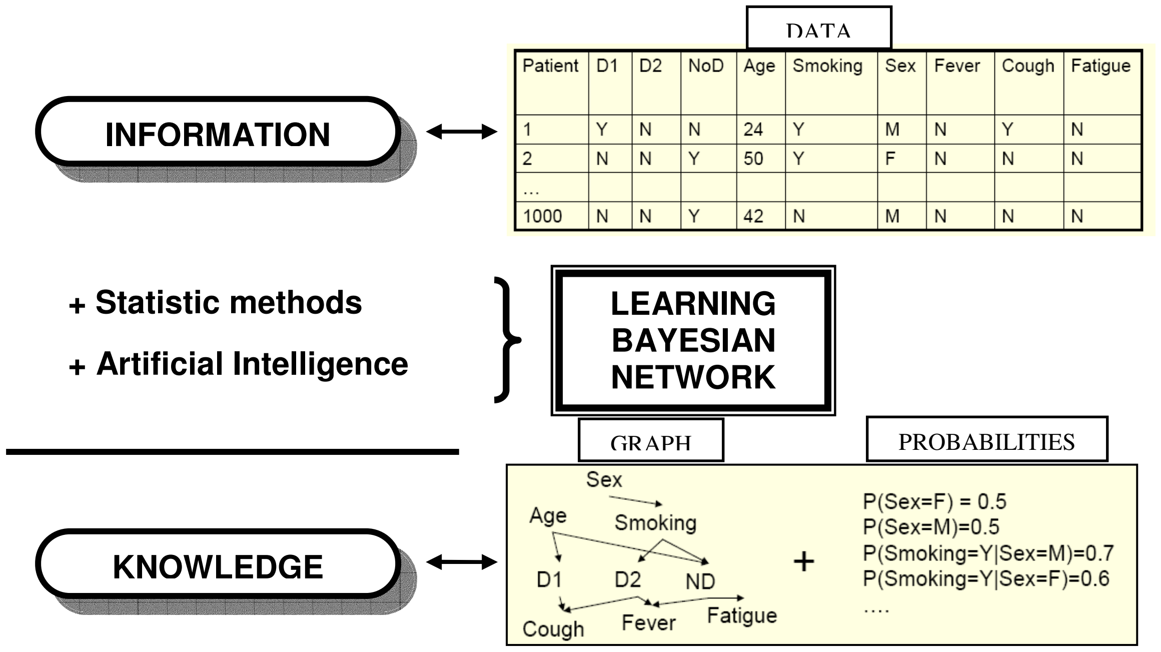
In my work I will only treat the task of learning the structure. This task can be divided in two sub-tasks:

- Learning the skeleton
- Orienting the edges

This paper focuses on the sub-task of learning the skeleton because it is the NP-Hard part of the problem.

To make this idea more clear I will give an example. In the graph of Fig. 1, every node represents a variable of the system and every arrow represents a causal dependency which has an associated probability. For the variables that do not have parents (like variable "Sex" in the example of Fig. 1) there are simple probabilities (P[Sex=F] =0,5), but for those who have parents (like variable "Smoking") there are conditional probabilities depending on the parent state (P[Smoking=Y/Sex=M]=0,7).

Once we have obtained the “knowledge” of how the system works, we can obtain the probability that a patient has a “cough”, “fever” or “fatigue”, by asking him some inputs such as “sex”, “age”, etc.



Patient	D1	D2	NoD	Age	Smoking	Sex	Fever	Cough	Fatigue
1	Y	N	N	24	Y	M	N	Y	N
2	N	N	Y	50	Y	F	N	N	N
…									
1000	N	N	Y	42	N	M	N	N	N

- Fig. 1 – Example for learning bayesian networks

The most interesting part of the Bayesian Networks is its intuitive face, because its output graph can be understood by everybody and not only by those who know Bayesian Networks. This gives a great advantage since everyone can correct and improve the graph.

2.3 ALGORITHMS

The first, however uninformed, idea to find the best network structure is the exploration of all possible graphs (giving each a score) in order to choose the graph with the best score. Robinson [Robinson, 1977] proved that r(n), the number of different structures for a bayesian network with “n” nodes, is given by the recurrence formula of equation 1.

$$r(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} r(n-i) = n^{2^{O(n)}} \quad \textbf{(1)}$$

This equation gives r(2) = 3, r(3) = 25, r(5) = 29281, r(10) = 4,2 x 10^{18}.

Since equation 1 is super exponential, it is impossible to perform an exhaustive search in a acceptable time period as soon as the node number exceeds 7 or 8.

To simplify the search, many network constructed algorithms have been developed. These algorithms can be grouped into two categories:

- One category of algorithms uses heuristic searching methods to construct a model and then evaluates it using a scoring method. This process continues until the score of the new model is not significantly better than the previous one. Different scoring criteria have been applied in these algorithms, such as, Bayesian scoring method [Cooper and Herskovits, 1992]; [Heckerman et al., 1994], entropy based method [Herskovits, 1991], and minimum description length method [Suzuki, 1996].

- The other category of algorithms constructs Bayesian networks by analyzing dependency relationships among nodes. The dependency relationships are measured by using some kind of conditional independence (CI) test. The algorithms described in [Spirtes et al., 1991]; [Wermuth and Lauritzen, 1983]; [Srinivas et al., 1990], [Cheng et al., 2001] and [Brown et al., 2005] are found in this group.

Both of these two categories of algorithms have their advantages and disadvantages: Generally, the first category of algorithms has less time complexity in the worst case scenario (when the underlying DAG is densely connected), however it may not find the best solution due to its heuristic nature. The second category of algorithms is usually asymptotically correct when the probability distribution of data satisfies certain assumptions, but as Cooper et al. pointed out in [Cooper and Herskovits, 1992], CI tests with large condition-sets may be unreliable unless the volume of data is enormous.

These algorithms are in the worst case exponential. This means that they need to calculate an exponential number of CI tests (for the first group) or of scores (for the second group) to obtain the structure of the bayesian network. However, there are two algorithms of the first group that have arrived to obtain polynomial complexity. There names are:

- PMMS (Polynomial Max-Min Skeleton)
- TPDA (Three Phase Dependency Analysis)

2.4 MARKOV EQUIVALENT SET AND COMPLETED-PDAGS

Definition 2. *Suppose we have a joint probability distribution P of the random variables in some set V and a DAG G = (V, E). We say that (G, P) satisfies the Markov condition if for each variable V Є X, {X} is conditionally independent of the set of all its no-descendents given the set of all its parents.*

Definition 3*. Two* ***dags*** *are said* ***equivalents*** *(noted ≡) if they imply the same set of conditional dependencies (they have the same joint distribution). The Markov equivalent classes set (named E) is defined as E = A/≡ where we named A the dags set.*

Definition 4. *An arc is said reversible if its reversion leads to a graph which is equivalent to the first one. The space of* ***Completed-PDAGs*** *(cpdags or also named essential graphs)* *is defined as the set of Partially Directed Acyclic Graphs (pdags) that have only undirected arcs and irreversible directed arcs.*

These three definitions are the bases to understand why a bayesian network can reduce so much the representation of a system. In a bayesian network not only the arcs gives information, but also its orientation and the absence of arcs gives a lot of information. It can be proved that in a faithful BN, an edge between X and Y exists if and only if there is no d-separating set **Z** such that X and Y are independent. To understand this concept, I will first show how the orientation of the edges gives as information and then I will explain what means "d-separating" two nodes and how it gives valuable information.

2.5 NODES ORIENTATION

For instance, as the Bayes rule gives

$$P(A,B,C) = P(A)P(B|A)P(C|B) = P(A|B)P(B)P(C|B) = P(A|B)P(B|C)P(C)$$

These structures are equivalents:

$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C & \equiv & A & B \arrow[l] \arrow[r] & C & \equiv & A & B \arrow[l] & C \arrow[l] \end{tikzcd}$$

Then, they can be schematised by a cpdag without ambiguities:

$$\begin{tikzcd} A \arrow[r, dash] & B \arrow[r, dash] & C \end{tikzcd}$$

However, they are not equivalent to a "collider" (also named "V-structure"). A collider is a node where two arcs meet at their endpoints:

$$\begin{tikzcd} A \arrow[r] & B & C \arrow[l] \end{tikzcd}$$

Where:

$$P(A,B,C) = P(A)P(B|A,C)P(C)$$

[Verma & Pearl, 1990] proved that dags are equivalent if, and only if, they have the same skeleton (i.e. the same edge support) and the same set of V-structures.

2.6 D-SEPARATION

TPDA and PMMS algorithms construct a bayesian networks by analyzing conditional independence relationships among nodes. To introduce this approach, we first review the concept of d-separation [Pearl, 1988], which plays an important role in these algorithms.

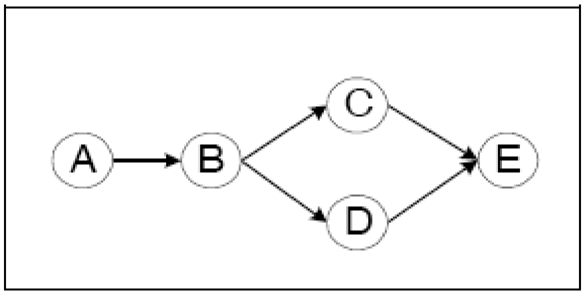
For any three disjoint node sets X, Y, and Z in a belief network, X is said to be d-separated from Y by Z if there is no active adjacency path between X and Y. An adjacency path is a path between two nodes without considering the directionality of the arcs. A path between X and Y is active given Z if:

(1) Every collider [Spirtes et al., 1996] in the path is in Z or has a descendant in Z
(2) Every other node in the path is outside Z.

In a belief network, if there is an arc from A to B, we say that A is a parent of B and B is a child of A. We also say that A is in the neighborhood of B and B is in the neighborhood of A.

To understand d-separation, I will use an analogy which has been elaborated by [Cheng et al., 2001]. They viewed a belief network as a network system of information channels, where each node was a valve that was either active or inactive and the valves were connected by noisy information channels. The information flow could pass an active valve but not an inactive one. When all the valves (nodes) on one adjacency path between two nodes were active, they said this path was **open**. If any valve in the path was inactive, they said the path was **closed**. When all paths between two nodes were **closed** given the statuses of a set of valves (nodes), they said the two nodes were d-separated by the set of nodes. This set of nodes was named as cut-set. The status of valves could be changed through the instantiation of a cut-set.

To clarify this explication I have included an example. In Figure 2, C-E-D is an adjacency path connecting C and D, even though the arcs are in different directions; we also say that E is a collider in the path C-E-D. Given empty evidence C and D are d-separated.



- Fig.2 – Example of d-separation

In the analogy of [Cheng et al., 2001], putting a node into the cut-set was equivalent to altering the status of the corresponding valves. For example, putting the collider E into the cut-set will open the path between C and D; while putting the non-collider B into the cut-set will close both the A-B-C-E and the A-B-D-E paths, thereby d-separating A and E.

2.7 FAITHFULNESS CONDITION

In a BN=(G; P):

$$\text{DSEP}_G(X;Y|Z) \Rightarrow \text{IND}(X;Y|Z)$$

In a faithful BN (G; P):

$$\text{DSEP}_G(X;Y|Z) \Leftrightarrow \text{IND}(X;Y|Z)$$

A BN=(G; P) satisfies the faithfulness condition (called faithful network) if the Markov Condition applied on G entails all and only the conditional independencies in P (Spirtes, Glymour, & Scheines 2000).

It can be proved that in a faithful BN, an edge between X and Y exists if and only if there is no d-separating set **Z** such that X and Y are independent (Spirtes, Glymour, & Scheines 2000). Algorithms following the constraint-based approach in BN learning estimate from data the conditional independencies and return only the edges which satisfy the above condition.

2.8 MUTUAL INFORMATION AND χ2 TEST OF INDEPENDENCE

The amount of information flow between two nodes can be measured by different methods. In particular, TPDA uses mutual information and PMMS uses χ2 test of independence.

The mutual information of two nodes (Xi; Xj) , is defined as:

$$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \qquad \textbf{(2)}$$

And the conditional mutual information is defined as:

$$I(X_i, X_j|C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) \log \frac{P(x_i, x_j|c)}{P(x_i|c), P(x_j|c)} \qquad \textbf{(3)}$$

where (X_i, X_j) are two nodes and C is a set of nodes.

TPDA algorithm uses conditional mutual information as CI tests to measure the average information between two nodes when the status of some valves are changed by the condition-set C. When I (X_i,X_j|C) is smaller than a certain threshold value ϵ (set to 0.01 in my experiments) , it is said that (X_i, X_j) are d-separated by the condition-set C, and they are conditionally independent.

PMMS algorithm is based on tests of conditional independence and measures of the strength of association between a pair of variables. To implement the test of independence - IND(X_i;X_j|$\mathbf{X}_k$) - it calculates the $\chi 2$ test as in Spirtes, Glymour and Scheines (2000), under the null hypothesis of the conditional independence holding.

The $\chi 2$ test returns a p-value that corresponds to the probability of falsely rejecting the null hypothesis given that it is true. If the p-value is less than a significance level α (set to 0.01 in my experiments) the null hypothesis is rejected. If the independence hypothesis cannot be rejected, it is accepted instead. A more detailed discussion on this use of independence tests can be found in Neapolitan (2003) pp. 593.

As a measure of association, PMMS algorithm uses the negative p-value returned by the $\chi 2$ test of independence: the smaller the p-value, the higher the association. To break the ties among equal p-values we used the $\chi 2$ statistic. Again, a p-value less than the 0.01 threshold was used to indicate a zero association.

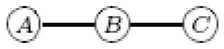
3. ALGORITHMS

In this section I will present TPDA and PMMS, which are the two algorithms that have been able to obtain polynomial complexity. For each algorithm I will present:

- The assumptions that have been made so as to apply these algorithms
- An explanation and an example of how each algorithm works

And then I will present a theoretical comparison between both algorithms.

It is important to note that PMMS is an algorithm that finds only the structure. If we want to obtain a direct graph, we will have to implement an orientation technique so as to define the edges. On the other hand, TPDA includes an orientation technique. As my work is centralise in the power each algorithm has to reconstruct the real structure, I decided to eliminate the orientation of the edges from TPDA algorithm and compare only the structures. That means that the arcs between variables will be always without orientation:



3.1 TPDA ALGORITHM

This algorithm was created by Cheng, Russell and Kelly in 2001. To explain it I use the explanation made by their authors in [Cheng et al., 2001]

3.1.1 ASSUMPTIONS

This algorithm can be used to obtain the structure of a bayesian network when:

- The records occur independently given the underlying probabilistic model of the data (that is, the dataset is "independent and identically distributed", iid)
- The cases in the data are drawn iid from a monotone DAG-faithful distribution
- The attributes of a table have discrete values and there are no missing values in any of the records
- The quantity of data is large enough for the conditional independent test used in the algorithm to be reliable.

Intuitively, the monotone DAG-faithful assumption requires that the conditional mutual information of two variables be a monotonic function of the active paths between the variables in the network structure: the more active open by a conditioning set, the greater the mutual information between the variables should be.

3.1.1 ALGORITHM

TPDA begins with a "drafting" phase, which produces an initial set of edges based on a simple test. The draft is a singly-connected graph (a graph without loops); found using (essentially) the Chow-Liu [1968] algorithm.

In the second phase or the "thickening", TPDA adds edges to the current graph when the pairs of nodes cannot be separated using a set of relevant CI tests. If the underlying model is DAG-faithful, the graph produced by this phase will contain all the edges of the underlying dependency model

The third or "thinning" phase corresponds to Step 3: here each of the edges is examined and removed if the two nodes of the edge are found to be conditionally independent.

TPDA then runs the OrientEdges procedure to define the essential arcs of the learned graph, to produce an essential graph. As I have previously mentioned, I am not considering the orientation of the edges because I will analyse only the skeleton.

The procedure of TPDA is:

Phase I: Drafting

1. Initiate a graph G(V,E) where V={all the nodes of a data set}, E={ }. Initiate an empty list L.

2. For each pair of nodes (v_i ; v_j) where v_i ,v_j Є V , compute mutual information I (v_i ; v_j) using equation (2). For all the pairs of nodes that have mutual information greater than a certain small value ϵ (that will be 0.01 for my implementations as it is suggest by the authors of TPDA), sort them by their mutual information and put these pairs of nodes into list L from large to small. Create a pointer p that points to the first pair of nodes in L.

3. Get the first two pairs of nodes of list L and remove them from it. Add the corresponding edges to E. Move the pointer p to the next pair of nodes.

4. Get the pair of nodes from L at the position of the pointer p. If there is no adjacency path between the two nodes, add the corresponding edge to E and remove this pair of nodes from L.

5. Move the pointer p to the next pair of nodes and go back to step 4 unless p is pointing to the end of L or G contains n-1 edges. (n is the number of nodes in G.)

In order to illustrate this algorithm's working mechanism, I use the example of [Cheng et al., 2001] where he uses a simple multi-connected network example taken from [Spirtes and al., 1996].

I have a database that has underlying Bayesian network as Figure 3.a and my task is to rediscover the underlying network structure from data. After step 2, I am able to obtain the mutual information of all 10 pair of nodes. I have:

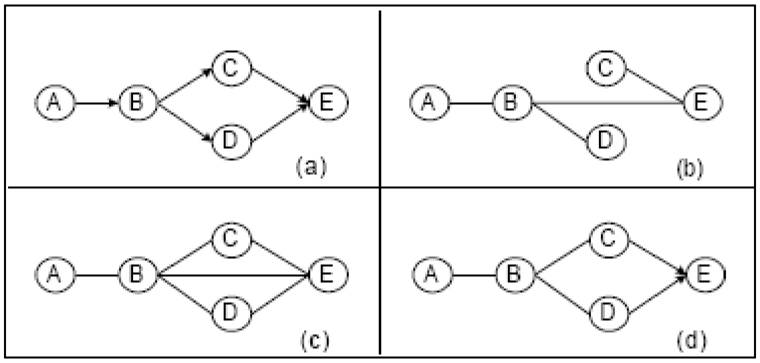
$$I(B,D) \geq I(C,E) \geq I(B,E) \geq I(A,B) \geq I(B,C) \geq I(C,D) \geq I(D,E) \geq I(A,D) \geq I(A,E) \geq I(A,C)$$

and all the mutual information is greater than ϵ, I can construct a draft shown in Figure 3.b after step 5. Please note that the order of mutual information between nodes can not be arbitrary. For example, from the information theory, I have:

$$I(A,C) < \mathrm{Min}(I(A,B),I(B,C))$$

This is also the reason why Phase one can construct a graph similar to the original graph to some extent. In fact, if the underlying graph is a singly connected graph, Phase one of this algorithm is essentially the algorithm of [Chow and Liu, 1968], and it guarantees the constructed network structure is the same as the original one.

In this example, (B,E) is wrongly added and (D,E) and (B,C) are missing because of the existing adjacency paths (D-B-E) and (B-E-C). The draft created in this phase is the base for next phase.



- Fig.3 - Example of TPDA

Phase II: Thickening

6. Move the pointer p to the first pair of node in L.

7. Get the pair of nodes from L at the position of the pointer p. Call procedure **try_to_separate_A (current graph, node1, node2)** to see if this pair of nodes can be separated in current graph. If so, go to next step; otherwise, connect the pair of nodes by adding a corresponding edge to E. (Procedure **try_to_separate_A** will be presented later in this subsection.)

8. Move the pointer p to the next pair of nodes and go back to step 7 unless p is pointing to the end of L.

In our example, the graph after Phase II is shown in Figure 3.c. Edge (B,C) and (D,E) are added because procedure **try_to_separate_A** cannot separate these pairs of nodes using CI tests. Edge (A,C) is not added because CI test can reveal that A and C are independent given block set {B}. Edge (A,D), (C,D) and (A,E) are not added for the same reason.

In this phase, the algorithm examines all pairs of nodes that have mutual information greater than ϵ and are not directly connected. An edge is not added only when the two nodes are independent given certain block set. However, it is possible that some edges are wrongly added in this phase. There are two reasons for this:

a) Some real edges may be still missing until the end of this phase, and these missing edges can prevent procedure **try_to_separate_A** from finding the correct condition-set.

b) As Procedure **try_to_separate_A** uses a heuristic method, it may not be able to find the correct condition-set for a special group of structures. (The detail is discussed later in this section.)

Phase III: Thinning

9. For each edge in E, if there are other paths besides this edge between the two nodes, remove this edge from E temporarily and call procedure **try_to_separate_A (current graph, node1, node2)**. If the two nodes are dependent, add this edge back to E; otherwise remove the edge permanently.

10. For each edge in E, if there are other paths besides this edge between the two nodes, remove this edge from E temporarily and call procedure **try_to_separate_B (current graph, node1, node2)**. If the two nodes are dependent, add this edge back to E; otherwise remove the edge permanently. (Procedure **try_to_separate_B** will be presented later in this subsection.)

11. Call procedure **orient_edges (current graph)**. (As I have said before, this procedure is not covered during my studies)

The 'thinned' graph of our example is shown in Figure 3.d, which has the same structure of the original graph. Edge (B,E) is removed because B and E are independent given {C,D}. If the underlying dependency model has a normal DAG-faithful probability distribution, the structure generated by this procedure is exactly the same as the structure of the underlying model. This phase can also orient edge (C,E) and edge (D,E) correctly.

Since procedure **try_to_separate_A** uses a heuristic method to find the condition-set, it may not always be able to separate two d-separated nodes. In order to guarantee that a correct structure can always be generated, we have to use a correct procedure **try_to_separate_B** at step 10 to re-examine the current edges.

Theoretically, we can use procedure **try_to_separate_B** to replace procedure **try_to_separate_A** in Phase II and remove step 9 in Phase III since they do the same thing. In practice, procedure **try_to_separate_A** usually uses fewer CI tests and requires smaller condition-sets. Therefore we try to avoid using procedure **try_to_separate_B** whenever it is possible.

PROCEDURE TRY_TO_SEPARATE_A (CURRENT GRAPH, NODE1, NODE2)

```
1: Find the neighbors of node1 and node2 that are on the adjacency paths between node1 and node2. Put them into two sets N1 and N2 respectively.
2: Remove the currently known child-nodes of node1 from N1 and child-nodes of node2 from N2.
3: If the cardinality of N1 is greater than that of N2, swap N1 and N2.
4: Use N1 as condition-set C.
5: Conduct a CI test using equation (3). Let $v = I(node1,node2|C)$. If $v < \epsilon$ , return ('d-separated').
6: If C contains only one node, go to step 8; otherwise, for each i, let $C_i = C \setminus \{the\ i\ th\ node\ of\ C\}$, $v_i = I(node1,node2|Ci)$. Find the smallest value $v_m$ of $v1$ , $v2$ ,…
7: If $vm < \epsilon$ , return ('d-separated'); otherwise, if $v_m > v$ go to step 8 else let $v = v_m$ , $C = C_m$ , go to step 6.
8: If N2 has not been used, use N2 as condition-set C and go to step 5; otherwise, return ('failed').
```

From the definition of Bayesian belief network, we know that if two nodes "a" and "b" in the network are not connected, they can be d-separated by the parent nodes of "b" which are in the paths between those two nodes. (We assume that node "a" appears earlier in the node ordering than "b"). Those parent nodes form a set P which is a subset of N1 or N2 of the above procedure. If node ordering is known, we can get P immediately and only one CI test is required to check if two nodes are d-separated. Since this information is usually not given, we have to use a group of CI tests to find such P. By assuming that removing a parent node of "b" will not increase the mutual information between "a" and "b", the above procedure try to find set P by identifying and removing the child-nodes and irrelevant nodes from N1 and N2 once a time using a group of computations and comparisons of conditional mutual information. However, this assumption may not be true when the underlying structure satisfies the following conditions:

- There exists at least one path from "a" to "b" through a child-node of "b" and this child-node is a collider on the path.
- In such paths, there are one or more colliders besides the child-node and all these colliders are the parents or ancestors of "b". In such structures, procedure **try_to_separate_A** may identify a parent node of "b" as a child-node of "b" and remove it erroneously. As a result, the procedure fails to separate two d-separated nodes. To deal with these structures, a correct procedure **try_to_separate_B** is introduced.

PROCEDURE TRY_TO_SEPARATE_B (CURRENT GRAPH, NODE1, NODE2)

```
1: Find the neighbors of node1 and node2 that are on the adjacency paths between node1 and node2. Put them into two sets N1 and N2 respectively.
2: Find the neighbors of the nodes in N1 that are on the adjacency paths between node1 and node2, and do not belong to N1. Put them into set N1'.
3: Find the neighbors of the nodes in N2 that are on the adjacency paths between node1 and node2, and do not belong to N2. Put them into set N2'.
4: If $|N1+N1'| < |N2+N2'|$ let set $C=N1+N1'$ else let $C=N2+N2'$.
5: Conduct a CI test using equation (3). Let $v = I(node1,node2|C)$. If $v < \epsilon$ , return ('d-separated').
6: Let $C'=C$. For each $i \in [1, |C|]$, let $Ci = C \setminus \{the\ i\ th\ node\ of\ C\}$, $vi = I(node1,node2|Ci)$. If $vi < \epsilon$ return ('d-separated') else if $v_i \leq v + \epsilon$ then $C'=C' \setminus \{the\ i\ th\ node\ of\ C\}$. ( $\epsilon$ is a small value )
7: If $|C'|<|C|$ then let $C=C'$, go to step 5; otherwise, return ('failed').
```

The major difference between procedure **try_to_separate_A** and **try_to_separate_B** is that instead of blocking the nodes of N1 or N2 the latter procedure also blocks nodes of set N1' or N2'. Since blocking two consecutive nodes in a path can always close the path, blocking set N1+N1' or N2+N2' can close all the paths that connect node1 and node2 through two or more nodes. The only open paths are those connect node1 and node2 through one collider. Under this circumstance, we can remove all the colliders that connect node1 and node2 without opening any previously closed paths. Thus, all paths between node1 and node2 in the underlying model can be closed.

In both procedure **try_to_separate_A** and procedure **try_to_separate_B**, we have to compare the mutual information on different condition-sets. If we do not use quantitative CI tests, we can not compare the results of different CI tests and therefore can not remove irrelevant nodes. By reducing the cardinality of the condition-set after each iteration, we can avoid to test on every subset of the initial condition-set and thus avoid exponential number of CI tests. However, qualitative CI test based algorithms have to carry out the test on every subset of C in order to separate two nodes, so the number of CI tests in such algorithms must be exponential in the worst case.

3.2 PMMS ALGORITHM

This algorithm was created by Brown, Tsamardinos and Aliferis in 2005. To explain it I use the explanation made by their authors in [Brown et al., 2005].

3.2.1 ASSUMPTIONS

The assumptions are the same as for TPDA algorithm; although the authors of PMMS in [Brown et al., 2005] claim that, in certain cases, PMMS can correctly reconstruct skeletons of networks that are not monotone DAG-faithful.

3.2.2 ALGORITHM

I will mark with PC_G^T the parents and children of T in the BN (G; P). This means that all the nodes with an edge to and from T will be in PC_G^T. This set will be unique for all G, always that (G; P) is a faithful Bayesian network to the same distribution P (Pearl 1988).

We define the minimum association of X and T relative to a feature subset **Z**, denoted as MINASSOC(X;T|Z), as the minimum association achieved between X and T over all subsets of **Z**:

$$\text{MINASSOC}(X;T|\mathbf{Z}) = \min_{\mathbf{S}\subseteq\mathbf{Z}} Assoc(X;T|\mathbf{S}) \quad \textbf{(2)}$$

ASSOC(X;T|S) is the association between two variables given a conditioning set. PMMS algorithm uses a statistically oriented test and the negative p-value returned by the χ^2 test of independence IND(X;T|S) (the smaller the p-value, the higher we consider the association between X and T) as in Spirtes, Glymour, & Scheines (2000).

The Polynomial Max-Min Skeleton algorithm (PMMS) is run on a dataset and returns the skeleton network. PMMS works by calling the Polynomial Max-Min Parents and Children algorithm (PMMPC) for each variable. PMMPC identifies an approximation of PC^T given a target node T and the data. Once the parents and children set has been discovered for each node, PMMS pieces together the identified edges into the network skeleton.

PMMPC(T,D), the main subroutine of the PMMS algorithm, discovers the PC^T using a two-phase scheme (shown in Fig. 5)[2].

PMMPC ALGORITHM

```
1: procedure PMMPC (T,D)
       %Phase I: Forward
2:     CPC = Ø
3:     repeat
4:         F = arg max_{X ∈ V} GREEDYMINASSOC(X;Y;{A,B};ASSOC(X;Y|Ø);Ø)
5:         assoc = max_{X ∈ V} GREEDYMINASSOC(X;Y;{A,B};ASSOC(X;Y|Ø);Ø)
6:         if assoc ≠ 0 then
7:             CPC = CPC ∪ F
8:     until CPC has not changed
       %Phase II: Backward
9:     for all X ∈ CPC do
10:        if GREEDYMINASSOC(X;Y;{A,B};ASSOC(X;Y|Ø);Ø) = 0 then
11:            CPC = CPC \ {X}
12:        return CPC
13: end procedure

14: function GREEDYMINASSOC(X,T,Z,minval,minarg)
15:     min = min_{S ∈ Z} ASSOC(X;T|minarg ∪{S})
16:     arg = arg min_{S ∈ Z} ASSOC(X;T|minarg ∪{S})
17:     if ((min < minval) AND (Z \ minarg = Ø)) then
18:         minval = GREEDYMINASSOC(X;T;Z \ minarg; min; minarg ∪ {arg})
19:     return minval
20: end function
```

Phase I: Forward

In this phase variables sequentially enter a candidate parents and children set of T, denoted as **CPC**, by use of the MAX-MIN HEURISTIC:

The heuristic is admissible in the sense that all variables with an edge to or from T and possibly more will eventually enter **CPC**. The intuitive justification for the heuristic is to select the variable that remains highly associated with T despite our best efforts to make the variable independent of T. Phase I stops when all remaining variables are independent of the target T given some subset of **CPC** (when the maximum minimum association reaches zero).

[2] *PMMPC* is a polynomial variant of the Max-Min Parents and Children (*MMPC*) algorithm (Tsamardinos, Aliferis, & Statnikov 2003). The *MMPC* algorithm can be created from PMMS algorithm by replacing the calls to GREEDYMINASSOC at lines 4, 5, and 10 with the function MINASSOC.

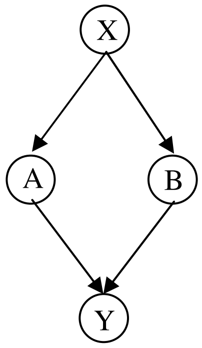
Conditioning on all subsets of **CPC** to identify the MINASSOC(X;T|CPC) requires an exponential number of calls to ASSOC. To get a polynomial number of calls, PMMPC greedily search for the subset S ⊆ CPC that achieves the minimum association between X and T conditioned over the subsets of **CPC**.

The function GREEDYMINASSOC(X,T,CPC,minval,minarg) starts with minval as the current estimate of the minimum and minarg as the current estimate of the minimizer S ⊆ CPC. Initially, minarg = Ø and minval = ASSOC(X;T| Ø). It then augments minarg by the member S of **CPC** that reduces the association ASSOC(X;T|minarg ∪{S}) the most. It continues in this fashion recursively until we condition on the full **CPC** or the minimum association achieved between X and T cannot be further reduced by augmenting minarg by a single variable.

For example, in the network structure of Fig. 6 the set Z={A,B} is a minimal d-separating set of X and Y . Assuming:

$$\text{ASSOC}(X;Y|\emptyset) \geq \text{ASSOC}(X;Y|\{A\}) \geq \text{ASSOC}(X;Y|\{A,B\})$$

then Z, a d-separating set, can be discovered in a greedy fashion.



- Fig. 4 - Example of PMMS

Phase II: Backward

In this phase PMMPC attempts to remove some of the false positives that may have entered in the first phase. The false positives are removed by testing IND(X;T|S) for some subset of the candidate parents and children set, S ⊆ **CPC**. If the independence holds, X is removed from **CPC**. The existence of an S ⊆ **CPC** for which IND(X;T|S) is approximated by testing whether GREEDYMINASSOC returns zero.

For example, in the network structure of Fig. 6 it is possible that X enters **CPC** before both A and B when the target is Y. Phase II, however, will remove X once both A and B are in

CPC and the test GREEDYMINASSOC(X;Y,{A,B},ASSOC(X;Y|Ø),Ø) returns zero (since ASSOC(X;Y|{A,B})=0).[3]

3.3 A COMPARISON BETWEEN TPDA AND PMMS ALGORITHM

TPDA and PMMS are similar at a fundamental level. Both have a forward phase where variables enter a candidate parent and children set for each node T. This set is the **CPC** in the PMMS algorithm and the set of neighbors of T in the current draft of the skeleton in TPDA. They also both have a backward phase where variables are removed from this set. During these phases, both algorithms attempt to discover a set **Z** such that ASSOC(X;T|Z)=0 for every X considered.

A major difference between the algorithms, however, is that PMMS builds up **Z** starting from the empty set. In contrast, TPDA starts by conditioning on the full set of the candidate parents and children and removing nodes from this set to reach **Z**.

When the available sample is not enough to condition on the full set of the parents and children set of T, I expect TPDA's strategy to fail to accurately estimate the conditional mutual information between T and any other node. On the other hand PMMS starts with the smallest conditioning set possible – the empty set – and proceeds with conditioning on larger sets only when the sample allows so. Thus, I expect PMMS to better reconstruct the skeleton when the available sample is low relative to the parent and children set sizes.

On the other hand, TPDA's strategy will pay off for relatively large sample sizes. For example, suppose that X and T can be d-separated by $\mathbf{S} \subseteq \mathbf{Z}$, where **Z** is the current estimate of PC_T . If the available sample is enough to condition on **Z** it may be possible that IND(X;T|Z), which TPDA will discover with a single call to mutual information. In contrast, PMMS has to perform at least |S| calls to ASSOC to identify **S**.

4. TEST

4.1 METHODOLOGY

The methodology I have chosen to implement these algorithms is:

1. choose a belief network (a network I already know)
2. generate randomly a data base from that network (taking into account the table of probabilities of the belief network)

[3] To avoid some false positives in PMMS it is possible to remove additional arcs by checking whether the symmetry $X \in PC_Y \Leftrightarrow Y \in PC_X$ holds (for more details see Tsamardinos, Brown, & Aliferis 2005).

3. learn the skeleton of the bayesian network from the data base (only with the information of the data base I have just generated)
4. analyse the quality of the skeleton I have found

4.2 SOFTWARE

There are many types of software that deals with Bayesian networks, for example:

- Hugin [Andersen et al., 1989]
- Netica [Norsys, 2003]
- Bayesia Lab [Munteanu et al., 2001]
- TETRAD [Scheines et al., 1994]
- DEAL [Bøttcher & Dethlefsen, 2003]
- LibB [LibB]
- Matlab: FullBNT Network [Murphy, 2001a]
- Matlab: SLP – BNT [Leray et al., 2003].
- Matlab: Causal Explorer [Aliferis et al.,2005]

The algorithms I am comparing are very recent; therefore they are may not be readily available in today's software. We can find some preliminary works on different softwares but not always the definitive algorithms. For my experiments, I have used:

- Matlab: FullBNT Network [Murphy, 2001a]
- Matlab: SLP – BNT [Leray et al., 2003].
- Matlab: Causal Explorer [Aliferis et al.,2005]
- Some preliminary developments that have been made in PRISMa laboratory [PRISMa]

I chose these softwares because the code source is open. This means I can see, modify and create the code of the software. However, I have to mention that the code source of Causal Explorer is not open but I can make a call to the algorithms that have been developed there. I think is also relevant to say that the creators of Causal Explorer toolbox are also the creators of PMMS although there is not yet available this algorithm in the toolbox.

As TPDA is an algorithm from 2001, there are two Matlab toolboxs that present preliminaries softwares for him:

- SLP – BNT (2003)
- Causal Explorer (2005)

On the other hand, PMMS is an algorithm from 2005 so there does not exist any Matlab toolbox that implements this algorithm. However, there exist a preliminary development of PRISMa laboratory where they have started to make a Matlab file of PMMS.

With all this tools I have satisfactory implemented TPDA and PMMS so as to learn the skeleton of the ASIA and ALARM networks. I have to mention that the SLP – BNT software for TPDA has a little variation that utilise the χ^2 test of independence instead of the mutual information.

4.3 NETWORKS

There are two things that I considered important so as to analyse the power of the algorithms:

- The complexity of the network they have to analyse
- The number of data they have

It is easy to see that as the network has more variables and arcs, the complexity of the networks augments. It is also intuitive that if we have more data it is more probable that the algorithm will find the real skeleton.

So as to take into account these two things, I decided to use two networks with different amount of data: 250, 500, 1.000, 2.000, 5.000, 10.000, 15.000 and 20.000.

The networks I chose are:

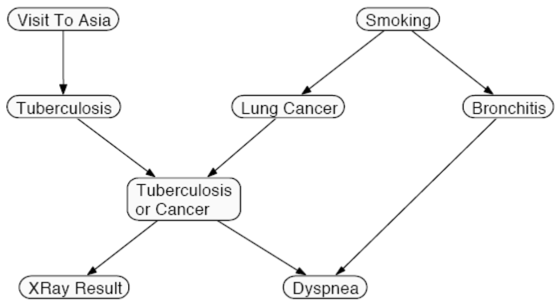
- ASIA: 8 nodes (each with 2 values); 8 arcs; 36 total parameters (http://www.norsys.com/netlib/Asia.dnet)
- ALARM: 37 nodes (each with 2, 3 or 4 values); 46 arcs; 509 total parameters [Beinlich et al., 1989]

The first network is from a simple fictitious medical domain and the second is from a moderate complex real-world domain.

4.3.1 ASIA NETWORK

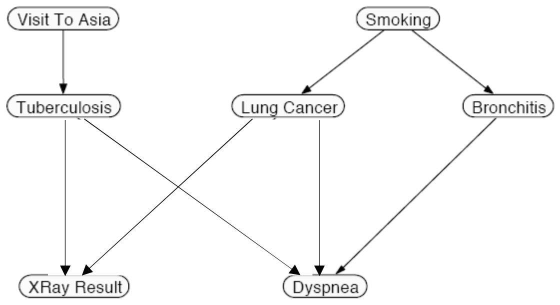
This network is a very small Bayesian network for a fictitious medical domain, relating whether a patient has tuberculosis, lung cancer or bronchitis, to their X-ray, dyspnea, visit-to-Asia and smoking status. The structure of this network is shown in Fig. 5.

I use this simple Bayesian network to show the performance of the algorithms on small domains.



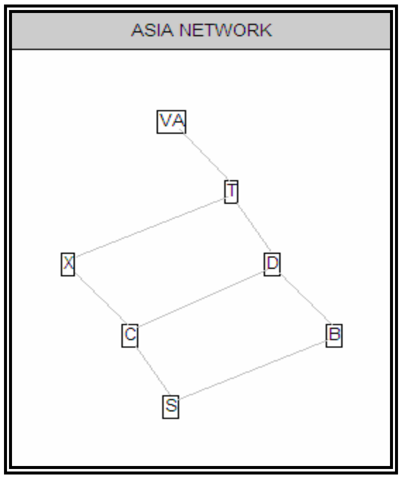
- Fig. 5 - ASIA network

The variable "Tuberculosis or Cancer" is not really a variable; it is an intermediate organizer for the graph. For this reason, it causes some problems to the algorithms and I decided to eliminate from the network. The new ASIA network is shown in Fig. 6.



- Fig. 6 – ASIA network (corrected)

As I have already said, I will not take into account the orientation of the edges in this paper. When I take out the orientation of the edge and I reorganize the variables distribution, I have the graph of Fig. 7.

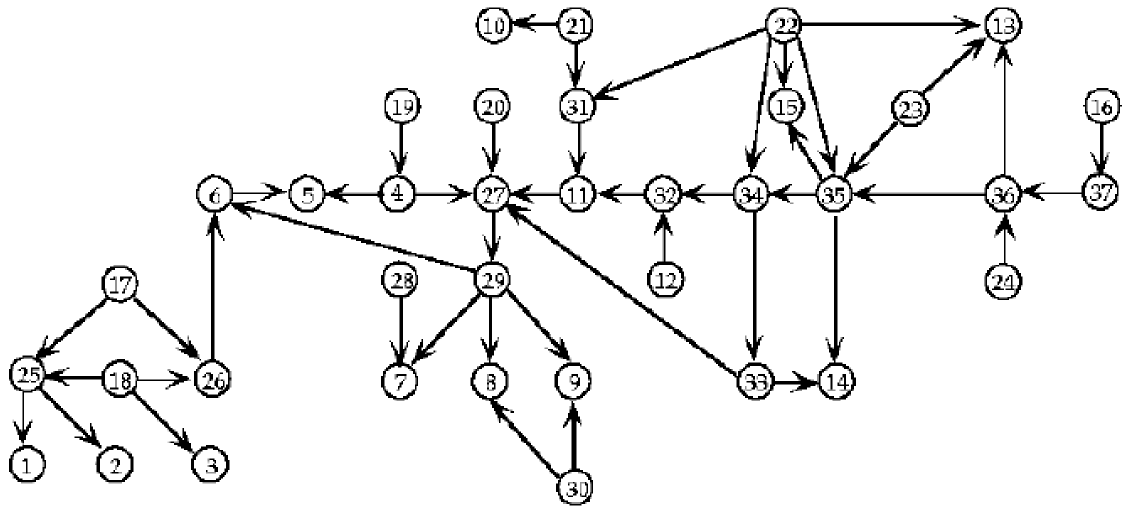


- Fig. 7 – ASIA network (corrected and without orientation) where I have abbreviated the variables as:

➢ VisitToAsia	→ VA
➢ Smoking	→ S
➢ Tuberculosis	→ T
➢ LungCancer	→ C
➢ Bronchitis	→ B
➢ Xray	→ X
➢ Dyspnea	→ D

4.3.2 ALARM NETWORK

ALARM, which stands for 'A Logical Alarm Reduction Mechanism', is a medical diagnostic system for patient monitoring, which includes nodes for 8 diagnoses, 16 findings and 13 intermediate variables [Beinlich et al., 1989]. Each variable has two to four possible values. The network structure is shown in Figure 19.



- Fig. 17 - ALARM network

The ALARM network is the most widely used benchmark in this area and many researchers have used it to evaluate their algorithms.

4.4 COMPARING SKELETONS

To compare the skeleton found with the true skeleton I have measured both the number of arcs that are missing and those that are in excess between them. This means that if both values are equal, I have found the original structure.

These measurements do not take into account the data that was created (because it compares it directly with the original belief network), so I have also have to measure the BIC score (Bayesian Information Criterion) of the new skeleton to see the correlation with the data base I have given. In all the sub-databases, the BIC score was larger when the addition of the measures was smaller giving us the information that the generated databases were coherent with the original belief network. The results of the BIC-score are shown in Annexe B.

4.4 RESULTS

4.4.1 NOTATION

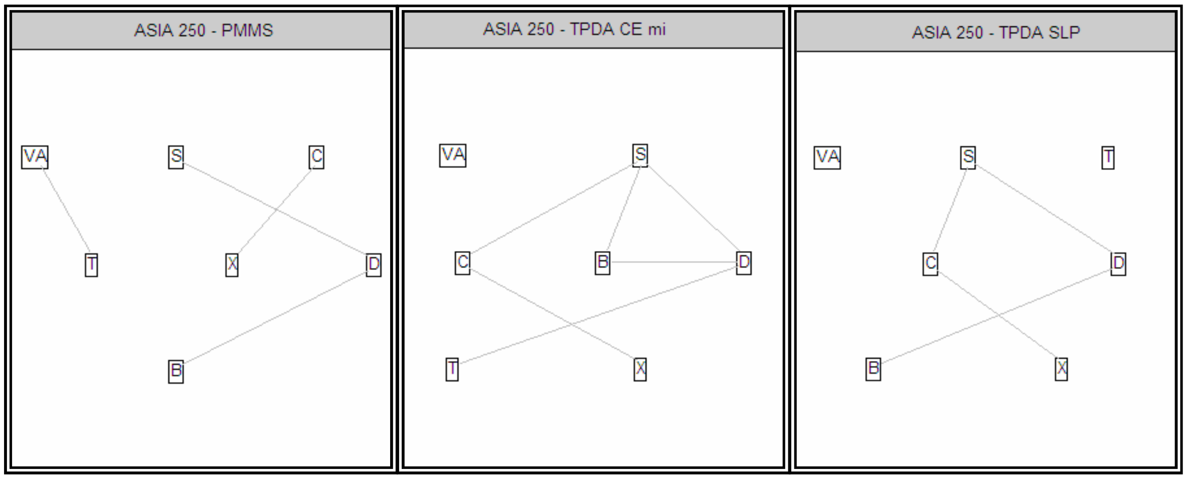
I used the following names to identify the different algorithms:

- PMMS: Is the PMMS algorithm implemented in base of some preliminaries developments of a Matlab file made by PRISMa laboratory.
- TPDA_CE_mi: Is the TPDA algorithm implemented with Causal Explorer and which uses “mutual information” to know the relation between variables.
- TPDA_SLP: Is the TPDA algorithm implemented with SLP – BNT and which has a little modification from the original TPDA algorithm. This algorithm uses “χ^2 test of independence” instead of “mutual information” to know the relation between variables.

In the next sections I will present the results of the 16 sub-database (8 for each network). The only difference between the sub-databases of a network is the number of data entries (250, 500, 1.000, 2.000, 5.000, 10.000, 15.000 and 20.000). It is important to remember that the values are not comparable between different sub-databases, but they are comparable between the different algorithms for the same sub-database.

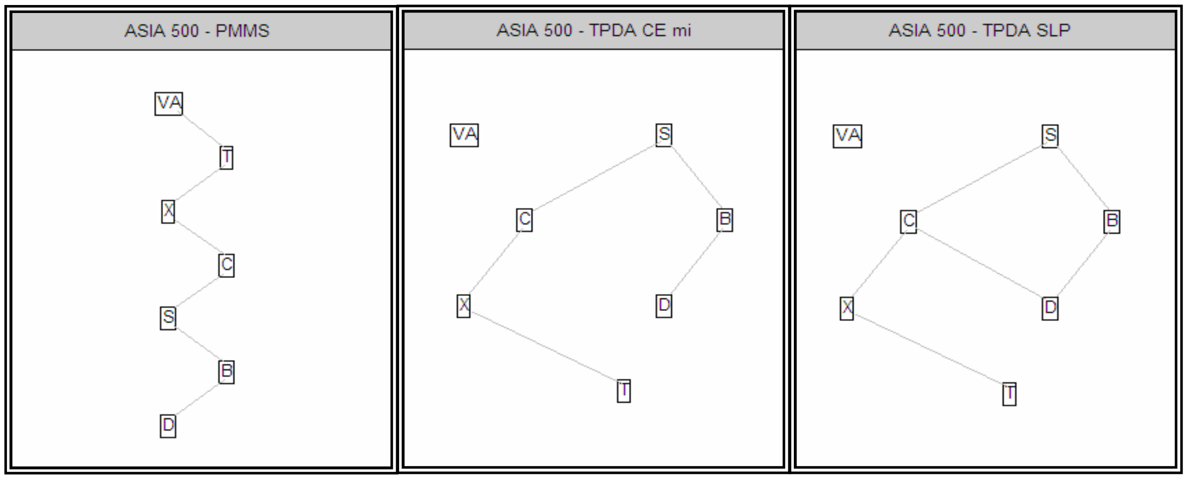
4.4.2 ASIA NETWORK

The graphs of the structures found are:

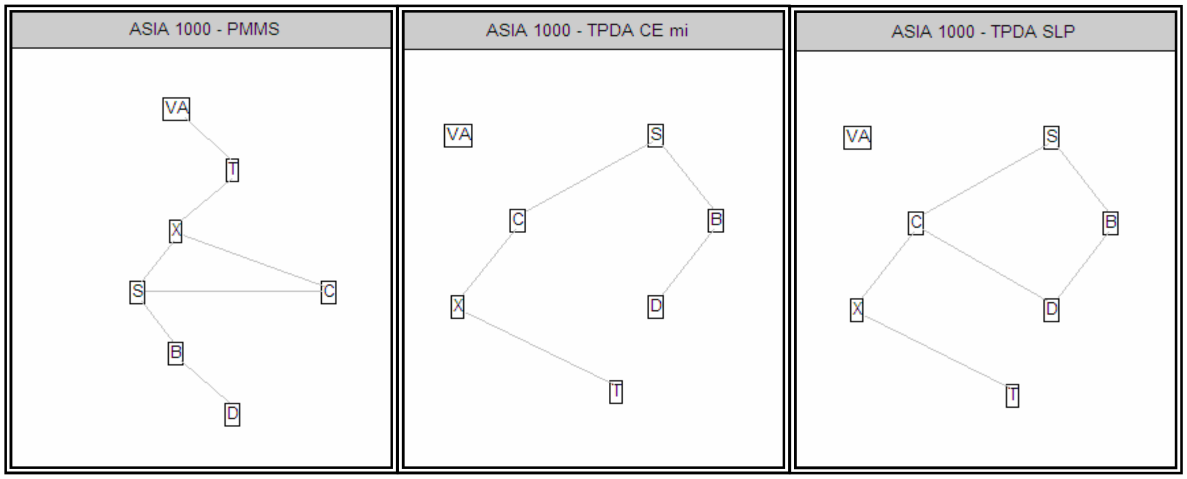


- Fig.8 - ASIA network with 250 data entries

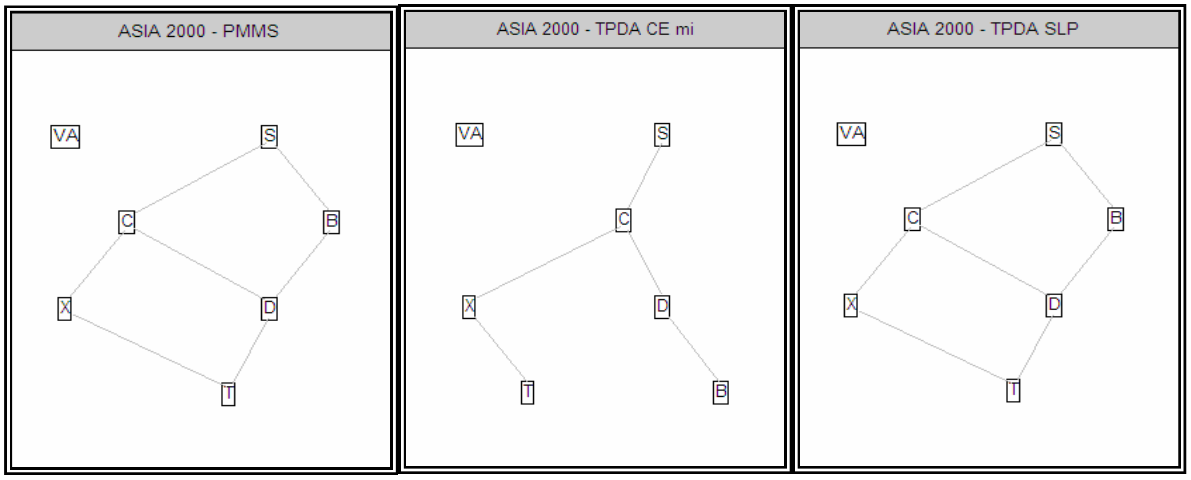
With only 250 data entries is quite difficult to identify the skeleton, even the simple ASIA network is quite far from the real structure. This sub- database only want to show that it is necessary a good amount of data for learning the skeleton of a bayesian network.



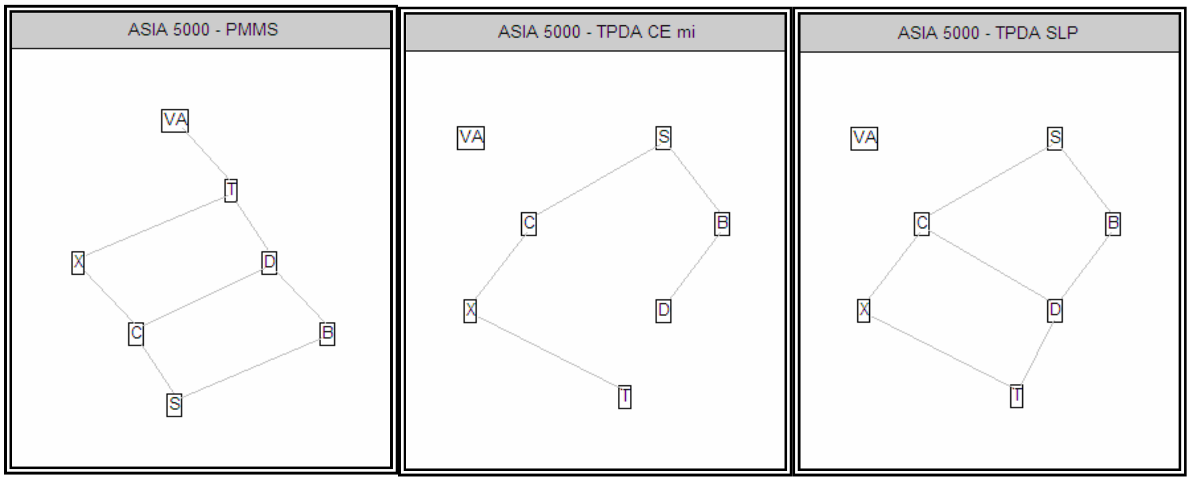
- Fig.9 - ASIA network with 500 data entries



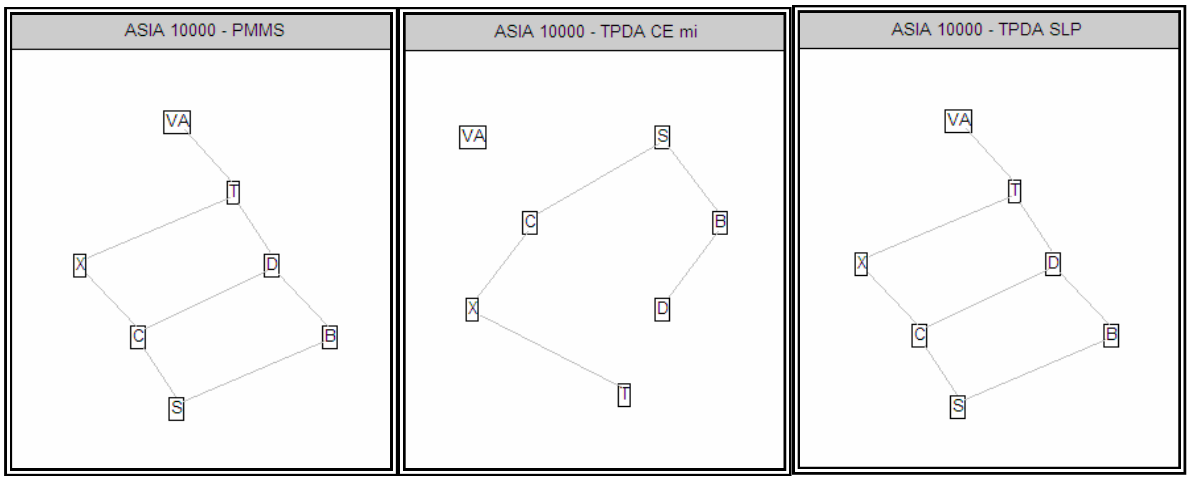
- Fig.10 - ASIA network with 1000 data entries



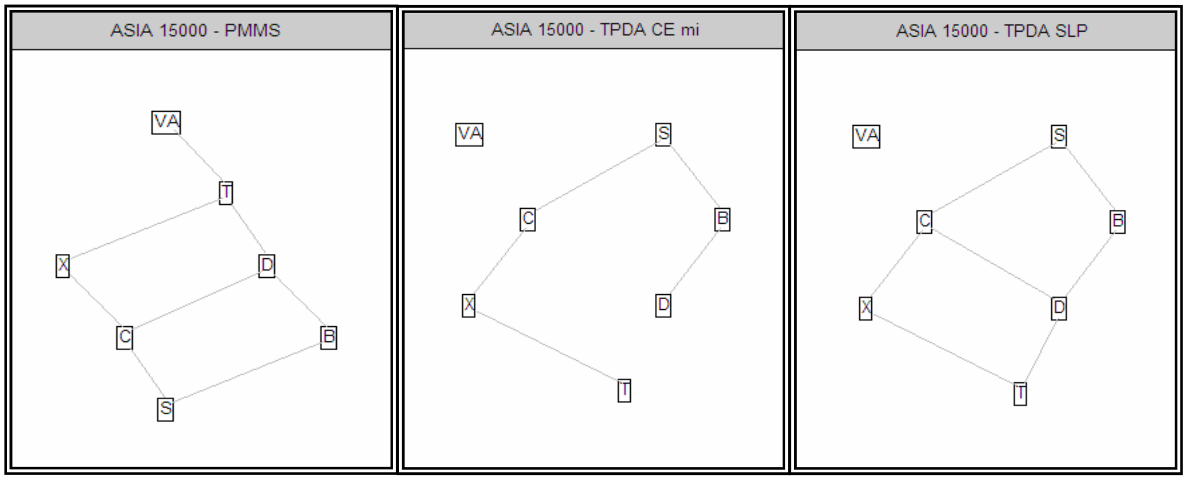
- Fig.11 - ASIA network with 2000 data entries



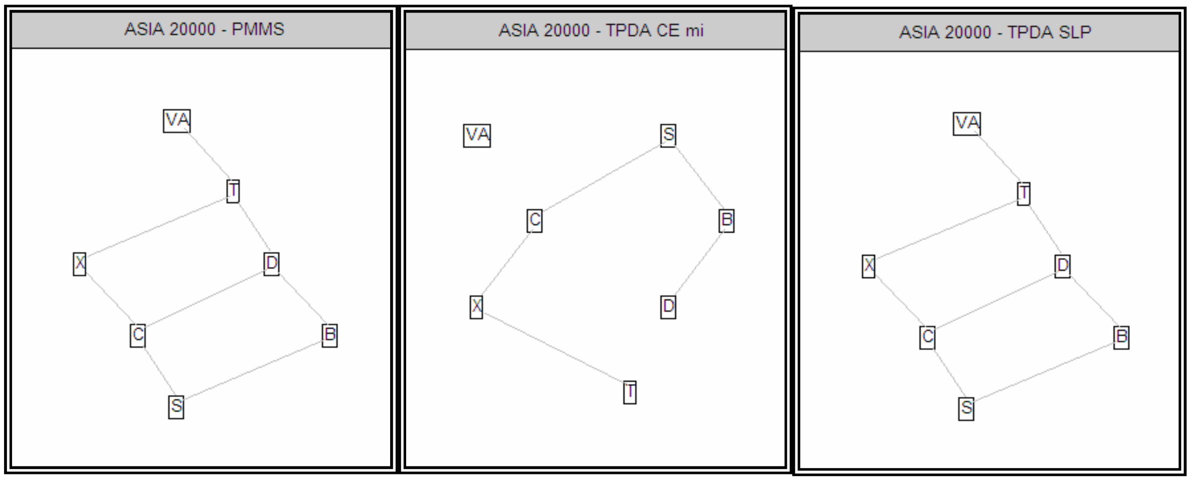
- Fig.12 - ASIA network with 5000 data entries



- Fig.13 - ASIA network with 10000 data entries



- Fig.14 - ASIA network with 15000 data entries

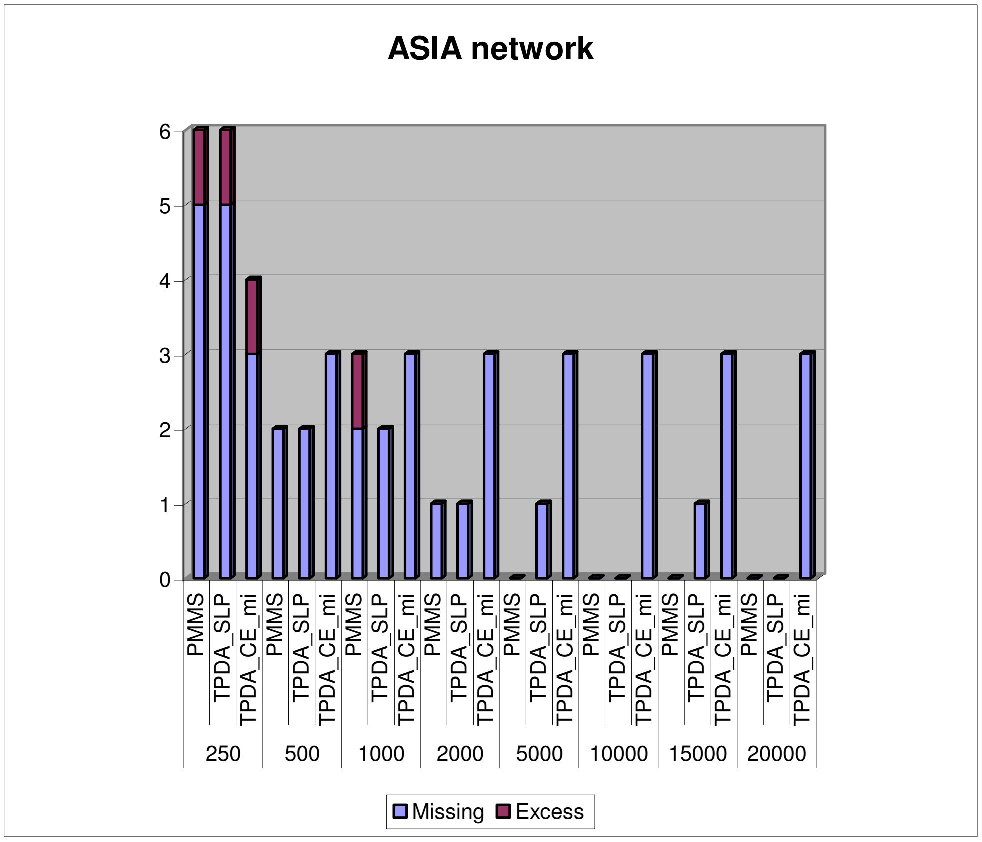


- Fig.15 - ASIA network with 20000 data entries

In Fig. 9 we start seeing the original skeleton, but it is still far away from the real skeleton. In the following graphs we see how they evolve just until they found the original graph. With 5.000 data entries, the PMMS algorithm is the first one in obtaining the real original skeleton (see Fig.12). The _SLP algorithm obtains the real original skeleton in the next sub-database (see Fig.13). From now over, augmenting the amount of data will not increase the quality of the found skeleton and TPDA_CE_mi will never found the correct skeleton.

We can see that there are some variables that are difficult to identify. The most evident case is the variable VA (Visit to Asia). It is easy to imagine that there are not so many cases of persons visiting Asia and that is why the algorithms have difficulties to identify this relation (the probability for a person to visit Asia in our belief network is only 0,01) . We can see that TPDA_CE_mi has never arrived to see this relation although, the others algorithms had arrived when the databases were big. On the other hand, PMMS has practically always found this relation (only in the 2000 sub database it could not found it).

In the following graph I will present the number of arcs missing and in excess that were found by the 3 algorithms in each sub-database:



- Fig.16 - Missing and excess arcs in ASIA network tests

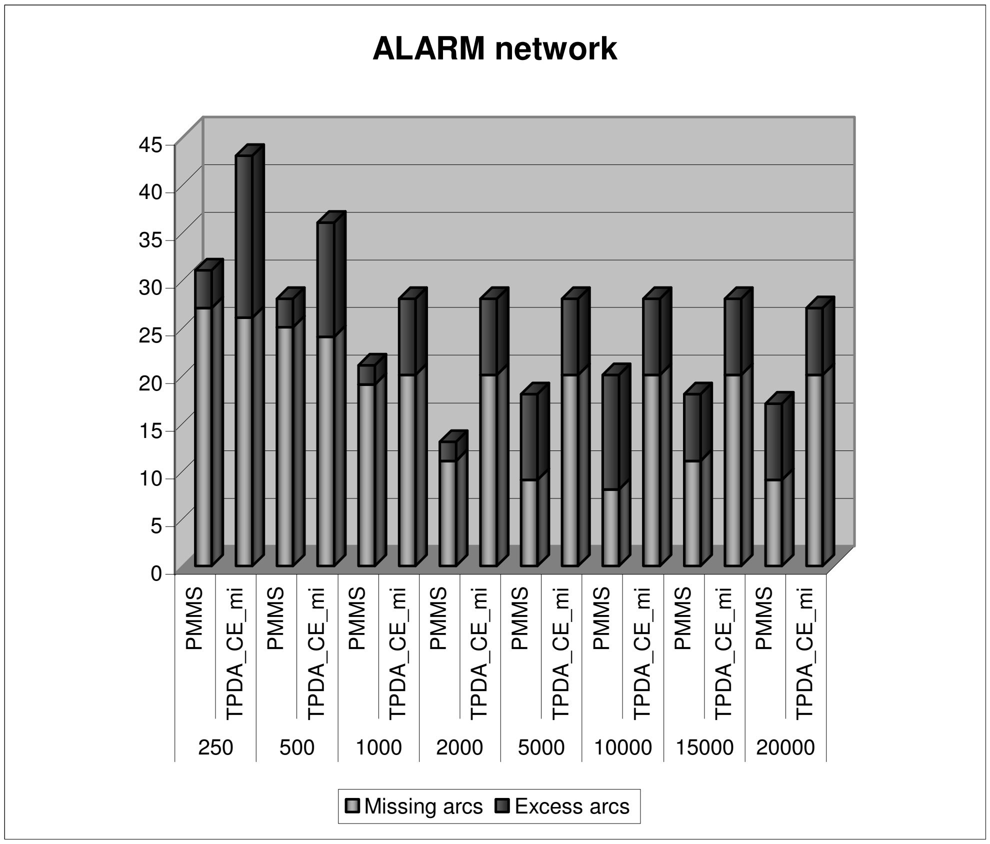
It is quite easy to conclude that for the ASIA network:

- The algorithms practically do not make mistakes of arcs excess, which means that the errors are due to some variables they do not arrive to see as related (arc missing).
- PMMS is the algorithm that obtains the corrected skeleton with fewer amounts of data (5000 data entries).
- PMMS and TPDA_SLP generally gives better results than TPDA_CE_mi
- The number of total mistakes for TPDA_CE_mi diminishes when augmenting the number of data entries, but it is stabilised after 500 data entries. Moreover, the number of missing arcs is always the same; the only difference is that with 250 entries it also gives an excess arc.
- The number of total mistakes for PMMS and TPDA_SLP diminishes when augmenting the number of data entries
- The number of mistakes of PMMS and TPDA_SLP are practically the same; although some times PMMS has better results.

4.4.3 ALARM NETWORK

In the ASIA network, I have made the graphs of the skeleton the algorithms have found because they can be easily understood by a quick view. I did not make the graph for the ALARM network because the resulting graph were so complicate that they were quite difficult to understands and they did not give me any additional information.

In the following graph I will present the distance of the 2 algorithms for each sub-database. As I have already said, the TPDA_SLP algorithm is a preliminary development and does not arrive to learn the structure of the ALARM network. That is the reason why it is not present in the graph.

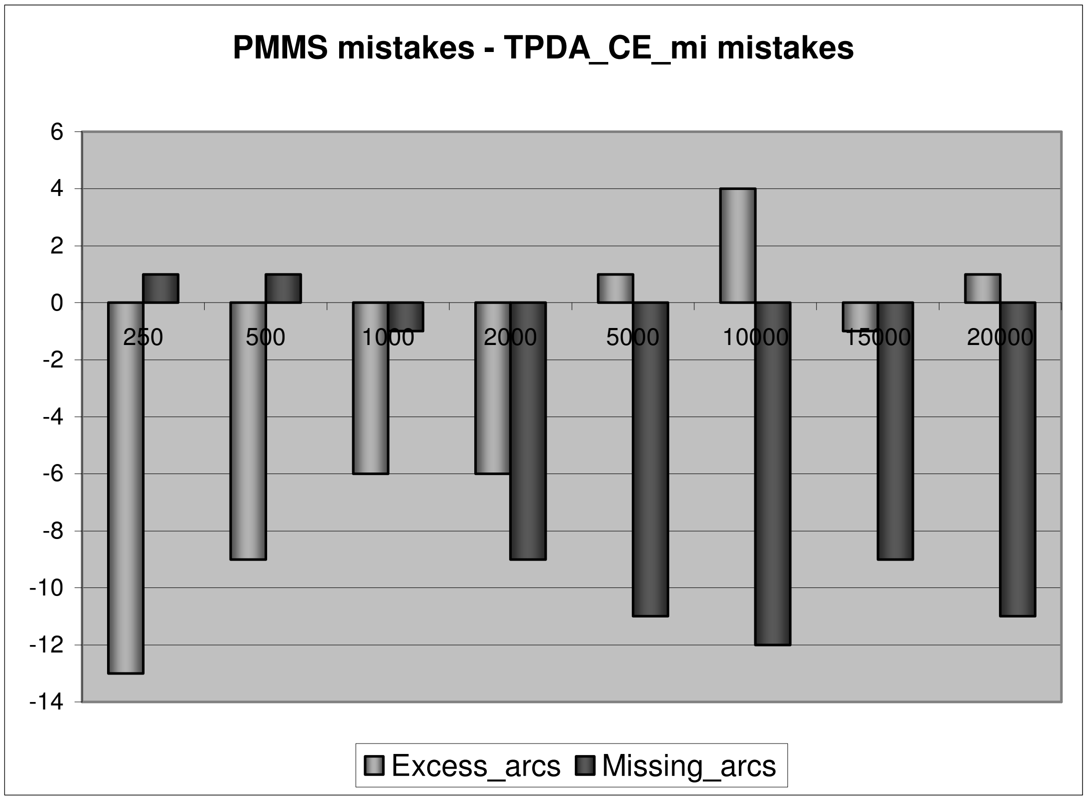


- Fig.18 - Missing and excess arcs in ALARM network tests

It is quite easy to conclude that for the ALARM network:

- The algorithms have missed more arcs than the ones that they have exceed
- None of the proposed algorithms arrive to obtain the corrected skeleton
- PMMS always has less total mistakes than TPDA_CE_mi
- The number of total mistakes for PMMS diminishes when augmenting the number of data entries just until 2000 entries. Then it gets more mistakes when augmenting the number of entries.
- The number of total mistakes for TPDA_CE_mi diminishes when augmenting the number of data entries, but it is stabilised after 1000 data entries.

To compare which algorithms makes more errors of missing and which one more of excess I have made another graph where I show the difference between the mistakes of PMMS and the ones of TPDA_CE_mi:



- Fig.19 - Difference between PMMS mistakes and TPDA_CE_mi mistakes

In the graph we can see that:

- TPDA_CE_mi makes more excess mistakes than PMMS with low amount of data. This tendency is reverse when the data is bigger than 5000
- PMMS makes more missing mistakes than TPDA_CE_mi with low amount of data. This tendency is reverse when the data is bigger than 1000

5. DISCUSSION AND CONCLUSION

In this paper I have explained the differences between PMMS and TPDA (the two algorithms that have arrived to obtain polynomial complexity for learning bayesian networks). In section 4, I have tested them so as to learn the structure of two belief bayesian network: The ASIA network, which is a simple fictitious medical problem; and the ALARM network, which is a moderate complex real-world problem. For testing these algorithms I have used different Matlab applications.

The results obtained for learning the skeletons of these two networks were better with the PMMS algorithm. Nevertheless, a little variation on the TPDA algorithm produces practically the same results of PMMS in the ASIA network. This variation consists in using the "χ^2 test of independence" (like in PMMS) instead of "mutual information"(like in standard TPDA) to know the relation between variables. The result is the algorithm I have presented by the name of TPDA_SLP. This algorithm implementation is not finished and actually it does not work for complex networks as ALARM, but the results in simple networks as ASIA are quite promising.

The final conclusions are:

- Actually the best option to learn the skeleton of a bayesian network with polynomial complexity is PMMS.
- PMMS outperform TPDA in small and big samples of data. However, it is important to see how it evolves the development of TPDA_SLP.
- The "χ^2 test of independence" have proved to obtain better results than the "mutual information" when learning bayesian networks with polynomial complexity.

6. ACKNOWLEDGEMENTS

I acknowledge Pr. Alexandre Aussem that has been my tutor through this project, as well as Zahra Kebaili and Sergio Rodrigues de Morais that have contributed with their comments and investigations.

REFERENCES

[Aliferis et al.,2005] C. Aliferis, I. Tsamardinos & A. Statnikov (2005) http://www.dsl-lab.org/index.html

[Andersen et al., 1989] Andersen S., Olesen K., Jensen F. & Jensen F. (1989). *Hugin - a shell for building bayesian belief universes for expert systems.* Proceedings of the 11th International Joint Conference on Artificial Intelligence, p. 1080–1085. http://www.hugin.com/.

[Beinlich *et al.*, 1989] Beinlich, I.A., Suermondt, H.J., Chavez, R.M. and Cooper, G.F., *The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks.* Proceedings of the Second European Conference on Artificial Intelligence in Medicine (pp.247-256), London, England, 1989.

[Bøttcher & Dethlefsen, 2003] Bøttcher S. G. & Dethlefsen C. (2003). *Deal: A package for learning bayesian networks.*

[Brown et al., 2005] Laura E. Brown, Ioannis and Tsamardinos; 2005. «*A Comparison of Novel and State-of-the-Art Polynomial Bayesian Network Learning Algorithms*». Vanderbilt University.

[Cheng et al., 2001] Jie Cheng , Russell Greiner and Jonathan Nelly; 2001. « *Learning Bayesian Networks from Data: An Information-Theory Based Approach* ». University of Alberta and University of Ulster.

[Chow and Liu, 1968] Chow, C.K. and Liu, C.N., Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14, 462-467, 1968.

[Cooper et Herskovits, 1992] Cooper, G.F., Herskovits, E., *A Bayesian Method for the induction of probabilistic networks from data*, Machine Learning, 9, 309-347, 1992.

[Heckerman *et al.*, 1994] Heckerman, D., Geiger, D. and Chickering, D.M., *Learning Bayesian networks: the combination of knowledge and statistical data*, Machine Learning Journal, 20(3), 1997.

[Herskovits, 1991] Herskovits, E., *Computer-based probabilistic network construction*, Doctoral dissertation, Medical information sciences, Stanford University, Stanford, CA, 1991.

[Jensen, 1996] Jensen F. V. (1996). *An introduction to Bayesian Networks.* Taylor and Francis, London, United Kingdom.

[Jordan, 1998] Jordan M. I. (1998). *Learning in Graphical Models*. The Netherlands: Kluwer Academic Publishers.

[Kim & Pearl, 1987] Kim J. & Pearl J. (1987). *Convice; a conversational inference consolidation engine*. IEEE Trans. on Systems, Man and Cybernetics, 17, 120–132.

[Lauritzen & Speigelhalter, 1988] Lauritzen S. & Speigelhalter D. (1988). *Local computations with probabilities on graphical structures and their application to expert systems.* Royal statistical Society B, 50, 157–224.

[Leray et al., 2003] Leray P., Guilmineau S., Noizet G., Francois O., Feasson E. & Minoc B. (2003). French BNT site. http://bnt.insa-rouen.fr/.

[LibB] http://www.cs.huji.ac.il/labs/compbio/LibB/programs.html

[Munteanu et al., 2001] Munteanu P., Jouffe L. & Wuillemin P. (2001). Bayesia lab.

[Murphy, 2001a] Murphy K. (2001a). The BayesNet Toolbox for Matlab, *Computing Science and Statistics: Proceedings of Interface*. http://bnt.sourceforge.net/

[Norsys, 2003] Norsys (2003). Netica.

[Pearl, 1988] Pearl, J., *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, 1988.

[PRISMa] http://prisma.insa-lyon.fr

[Réseaux Bayesians] Patrick Naïm, Pierre-Henri Wuillemin, Philippe Leray, Olivier Pourret y Anna Becker, 2004. «*Réseaux bayésiens*». 298 pages. Publisher Enrolles. ISBN 2-212-11137-1.

[Robinson, 1977] Robinson R. W. (1977). Counting unlabeled acyclic digraphs. In C. H. C. Little, Ed., Combinatorial Mathematics V, volume 622 of Lecture Notes in Mathematics, p. 28–43, Berlin: Springer.

[Scheines et al., 1994] Scheines R., Spirtes P., Glymour C. & Meek C. (1994). Tetrad ii: Tools for discovery. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Schwartz, 1978] Schwartz G. (1978). Estimating the dimension of a model. The Annals of Statistics, 6(2), 461–464.

[Spirtes *et al.*, 1991] Spirtes, P., Glymour, C. and Scheines, R., *An algorithm for fast recovery of sparse causal graphs*, Social Science Computer Review, 9, 62-72, 1991.

[Spirtes *et al.*, 1996] Spirtes, P., Glymour, C. and Scheines, R., *Causation, Prediction, and Search* (Book), http://hss.cmu.edu/html/departments/philosophy/TETRAD.BOOK/book.html,1996.

[Srinivas *et al.*, 1990] Srinivas, S. Russell, S. and Agogino, A., *Automated construction of sparse Bayesian networks from unstructured probabilistic models and domain information*, In Henrion, M., Shachter, R.D., Kanal, L.N. and Lemmer, J.F. (Eds.), *Uncertainty in artificial intelligence 5*, Amsterdam: North- Holland, 1990.

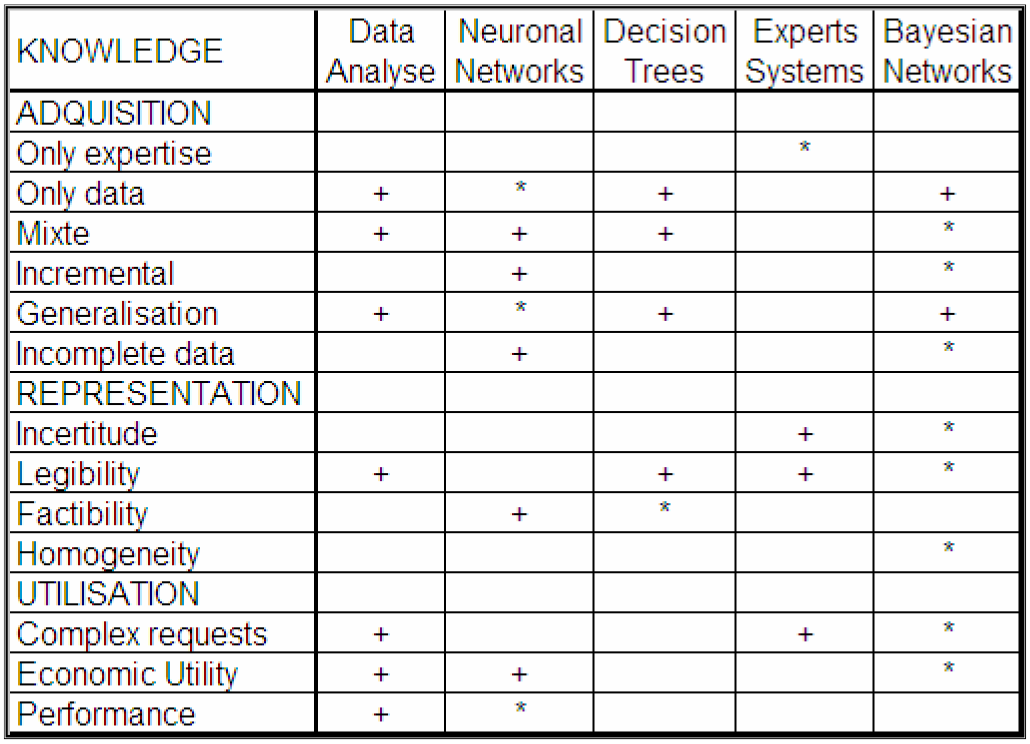
[Suzuki, 1996] Suzuki, J., *Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique,* Proceedings of the international conference on machine learning, Bari, Italy, 1996.

[Verma & Pearl, 1990] Verma T. & Pearl J. (1990). Equivalence and synthesis of causal models. In in Proceedings Sixth Conference on Uncertainty and Artificial Intelligence, San Francisco: Morgan Kaufmann.

[Wermuth and Lauritzen, 1983] Wermuth, N. and Lauritzen, S., *Graphical and recursive models for contingency tables*. Biometrika, 72, 537-552, 1983.

ANNEXE A: COMPARISON BETWEEN DIFERENT METODS FOR TRANSFORMING INFORMATION INTO KNOWLEDGE

Source: Table obtain from the page 165 of the book "*Réseaux Bayesiens*" [Réseaux Bayesians]



KNOWLEDGE	Data Analyse	Neuronal Networks	Decision Trees	Experts Systems	Bayesian Networks
ADQUISITION					
Only expertise				*	
Only data	+	*	+		+
Mixte	+	+	+		*
Incremental		+			*
Generalisation	+	*	+		+
Incomplete data		+			*
REPRESENTATION					
Incertitude				+	*
Legibility	+		+	+	*
Factibility		+	*		
Homogeneity					*
UTILISATION					
Complex requests	+			+	*
Economic Utility	+	+			*
Performance	+	*			

- Table 1 -

Reference:

+ : If the technique takes into account this problem or if it presents this advantage

* : If the technique is the best one in that characteristic

ANNEXE B: BIC SCORE

The BIC score derives from principles stated in [Schwartz, 1978] and has the following formulation:

$$BIC(\mathcal{B}, D) = \log \mathbb{P}(D|\mathcal{B}, \theta^{ML}) - \frac{1}{2} Dim(\mathcal{B}) \log N \qquad \textbf{(2)}$$

where D is the dataset, θ^{ML} are the parameter values obtained by likelihood maximisation, and where the network dimension Dim(B) is defined as follows:

As we need ri-1 parameters (probabilities) to describe the conditional probability distribution $P(X_i/Pa(X_i) = pa_i)$, where r_i is the size of X_i and pa_i is a specific value of X_i parents, we need $Dim(X_i, B)$ parameters to describe $P(X_i/Pa(X_i))$

$$Dim(X_i, \mathcal{B}) = (r_i - 1) q_i \quad \text{where} \quad q_i = \prod_{X_j \in Pa(X_i)} r_j \qquad \textbf{(3)}$$

And the bayesian network dimension Dim(B) is defined by

$$Dim(\mathcal{B}) = \sum_{i=1}^{n} Dim(X_i, \mathcal{B}) \qquad \textbf{(4)}$$

The BIC-score is the sum of a likelihood term and a penalty term which penalise complex networks. If two equivalent graphs have the same likelihood and the same complexity, the BIC-score will be equivalent (a score is said equivalent if it gives the same results for equivalent dags).

The BIC scores for the two networks I have implemented are:

ASIA network:

BIC score	250	500	1000	2000	5000	10000	15000	20000
PMMS	-619	-1218	-2318	-4526	-11234	-22572	-33724	-44810
TPDA_CE_mi	-611	-1214	-2307	-4643	-11381	-22818	-34115	-45367
TPDA_SLP	-612	-1211	-2282	-4526	-11231	-22572	-33722	-44810

- Table 2 -

ALARM netwrok:

BIC score	250	500	1000	2000	5000	10000	15000	20000
PMMS	-5319	-8661	-15655	-29634	-69261	-147290	-204760	-253640
TPDA_CE_mi	-6651	-9907	-17676	-32186	-74297	-141680	-210520	-277700

- Table 3 -

ANNEXE C: APPLICATIONS BASED ON BAYESIAN NETWORKS

In this annex I present a list of applications based on Bayesian networks that was presented in chapter 12 of Neapolitan book: "Learning Bayesian Networks". It includes applications in which structure was learned from data and ones in which the Bayesian network was constructed manually. The list is by no means meant to be exhaustive.

Academics

• The Learning Research and Development Center at the University of Pittsburgh developed Andes (www.pitt.edu/~vanlehn/andes.html), an intelligent tutoring system for physics. Andes infers a student's plan as the student works on a physics problem, and it assesses and tracks the student's domain knowledge over time. Andes is used by approximately 100 students/year.
• Royalty et al [2002] developed POET, which is an academic advising tool that models the evolution of a student's transcripts. Most of the variables represent course grades and take values from the set of grades plus the values "NotTaken" and "Withdrawn". This and related papers can be
found at www.cs.uky.edu/~goldsmit/papers/papers.html.

Biology

• Friedman et al [2000] developed a technique for learning causal relationships among genes by analyzing gene expression data. This technique is a result of the "Project for Using Bayesian Networks to Analyze Gene Expression," which is described at www.cs.huji.ac.il/labs/compbio/expression.
• Friedman et al [2002] developed a method for phylogenetic tree reconstruction.
The method is used in SEMPHY, which is a tool for maximum likelihood phylogenetic reconstruction. More on it can be found at www.cs.huji.ac.il/labs/compbio/semphy/.

Business and Finance

• Data Digest (www.data-digest.com) modeled and predicted customer behavior in a variety of business settings.
• The Bayesian Belief Network Application Group (www.soc.staffs.ac.uk/~cmtaa/bbnag.htm) developed applications in the financial sector. One application concerned the segmentation of a bank's customers. Business segmentation rules, which determine the classification of a bank's customers, had previously been implemented using an expert systems rulebased approach. This group developed a Bayesian network implementation of the rules. The developers say the Bayesian network was demonstrated to senior operational management within Barclays Bank, and these management personnel readily understood its reasoning. A second application concerned the assessment of risk in a loan applicant.

Capital Equipment

• Knowledge Industries, Inc. (KI) (www.kic.com) developed a relatively large number of applications during the 1990s. Most of them are used in internal applications by their licensees and are not publicly available. KI applications in capital equipment include locomotives, gas-turbine engines for aircraft and land-based power production, the space shuttle, and office equipment.

Causal Learning

• Applications to causal learning are discussed in [Spirtes et al, 1993, 2000].
• Causal learning applications also appear in [Glymour and Cooper, 1999].

Computer Games

• Valadares [2002] developed a computer game that models the evolution of a simulated world.

Computer Vision

• The Reading and Leeds Computer Vision Groups developed an integrated traffic and pedestrian model-based vision system. Information concerning this system can be found at www.cvg.cs.rdg.ac.uk/~imv.
• Huang et al [1994] analyzed freeway traffic using computer vision.
• Pham et al [2002] developed a face detection system.

Computer Hardware

• Intel Corporation (www.intel.com) developed a system for processor fault diagnosis. Specifically, given end-of-line tests on semi-conductor chips, it infers possible processing problems. They began developing their system in 1990 and, after many years of "evolution", they say it is now pretty stable. The network has three levels and a few hundred nodes. One difficulty they had was obtaining and tuning the prior probability values.
The newer parts of the diagnosis system are now being developed using a fuzzy-rule system, which they found to be easier to build and tune.

Computer Software

• Microsoft Research (research.microsoft.com) has developed a number of applications. Since 1995, Microsoft Office's AnswerWizard has used a naive-Bayesian network to select help topics based on queries. Also since 1995, there are about ten troubleshooters in Windows that use Bayesian networks. See [Heckerman et al, 1994].
• Burnell and Horvitz [1995] describe a system, which was developed by UT-Arlington and American Airlines (AA), for diagnosing problems with legacy software, specifically the Sabre airline reservation system used by AA. Given the information in a dump file, this

diagnostic system identifies which sequences of instructions may have led to the system error.

Data Mining

• Margaritis et al [2001] developed NetCube, a system for computing counts of records with desired characteristics from a database, which is a common task in the areas of decision support systems and data mining. The method can quickly compute counts from a database with billions of records. See www.cs.cmu.edu/~dmarg/Papers for this and related papers.

Medicine

• Knowledge Industries, Inc. (KI) (www.kic.com) developed a relatively large number of applications during the 1990s. Most of them are used in internal applications by their licensees and are not publicly available.
KI applications in medicine include sleep disorders, pathology, trauma care, hand and wrist evaluations, dermatology, and home-based health evaluations. They have the demonstration site www.Symptomedix.com, which is a site for the interactive diagnosis of headaches. It was designed and built to show the principles of operation of a Bayesian network in a medical application. It is medically correct for the domain of interest and has been tested in clinical application. The diagnostic system core was built with the KI DXpress Solution Series Software and has been widely used to demonstrate the use of Bayesian networks for diagnosis over the web.
• Heckerman et al [1992] describe Pathfinder, which is a system that assists community pathologists with the diagnosis of lymph node pathology.
Pathfinder has been integrated with videodiscs to form the commercial system Intellipath.
• Nicholson [1996] modeled the stepping patterns of the elderly to diagnose falls.
• Mani et al [1997] developed MENTOR, which is a system that predicts mental retardation in newborns.
• Herskovits and Dagner [1997] learned from data a system for assessing cervical spinal-cord trauma.
• Chevrolat et al [1998] modeled behavioral syndromes, in particular depression.
• Sakellaropoulos et al [1999] developed a system for the prognosis of head injuries.
• Onisko [2001] describes Hepar II, which is a system for diagnosing liver disorders.
• Ogunyemi at al [2002] developed TraumaSCAN, which assesses conditions arising from ballistic penetrating trauma to the chest and abdomen. It accomplishes this by integrating three-dimensional geometric reasoning about anatomic likelihood of injury with probabilistic reasoning about injury consequences.
• Galán et al [2002] created NasoNet, which is a system that performs diagnosis and prognosis of nasopharyngeal cancer (cancer concerning the nasal passages).

Natural Language Processing

• The University of Utah School of Medicine's Department of Medical Informatics developed SymText, which uses a Bayesian network to 1) represent semantic content; 2) relate words used to express concepts; (3) disambiguate constituent meaning and structure; 4) infer terms omitted due to ellipsis, errors, or context-dependent background knowledge; and 5) various other natural language processing tasks. The developers say the system is used constantly. So far four networks have been developed, each with 14 to 30 nodes, 3 to 4 layers, and containing an average of 1,000 probability values. Each network models a "context" of information targeted for extraction. Three networks exhibit a simple tree structure, while one uses multiple parents to model differences between positive and negated language patterns. The developers say the model has proven to be very valuable but carries two difficulties. First, the knowledge engineering tasks to create the network are costly and time consuming. Second, inference in the network carries a high computational cost. Methods are being explored for dealing with these issues. The developer say the model serves as an extremely robust backbone to the NLP engine.

Planning

• Dean and Wellman [1991] applied dynamic Bayesian networks to planning and control under uncertainty.
• Cozman and Krotkov [1996] developed quasi-Bayesian strategies for efficient plan generation.

Psychology

• Glymour [2001] discusses applications to cognitive psychology.

Reliability Analysis

• Torres-Toledano and Sucar [1998] developed a system for reliability analysis in power plants. This paper and related ones can be found at the site w3.mor.itesm.mx/~esucar/Proyectos/redes-bayes.html.
• The Centre for Software Reliability at Agena Ltd. (www.agena.co.uk) developed TRACS (Transport Reliability Assessment and Calculation System), which is a tool for predicting the reliability of military vehicles. The tool is used by the United Kingdom's Defense Research and Evaluation Agency (DERA) to assess vehicle reliability at all stages of the design and development life-cycle. The TRACS tool is in daily use and is being applied by DERA to help solve the following problems:
1. Identify the most likely top vehicles from a number of tenders before prototype development and testing begins.
2. Calculate reliability of future high-technology concept vehicles at the requirements stage.
3. Reduce the amount of resources devoted to testing vehicles on test tracks.
4. Model the effects of poor quality design and manufacturing processes on vehicle reliability.
5. Identify likely causes of unreliability and perform "what-if?" analyses to investigate the most profitable process improvements.
The TRACS tool is built on a modular architecture consisting of the following five major Bayesian networks:

1. An updating network used to predict the reliability of sub-systems based on failure data from historically similar sub-systems.
2. A recursive network used to coalesce sub-system reliability probability distributions in order to achieve a vehicle level prediction.
3. A design quality network used to estimate design unreliability caused by poor quality design processes.
4. Amanufacturing quality network used to estimate unreliability caused by poor quality manufacturing processes.
5. A vehicle testing network that uses failure date gained from vehicle testing to infer vehicle reliability.

The TRACS tool can model vehicles with an arbitrarily large number of sub-systems. Each sub-system network consists of over 1 million state combinations generated using a hierarchical Bayesian model with standard statistical distributions. The design and manufacturing quality networks contain 35 nodes, many of which have conditional probability distributions elicited directly from DERA engineering experts.
The TRACS tool was built using the SERENE tool and the Hugin API (www.hugin.dk), and it was written in VB using the MSAccess database engine. The SERENE method (www.hugin.dk/serene) was used to develop the Bayesian network structures and generate the conditional probability tables. A full description of the TRACS tool can be found at www.agena.co.uk/tracs/index.html.

Scheduling

• MITRE Corporation (www.mitre.org) developed a system for real-time weapons scheduling for ship self defense. Used by the United States Navy (NSWC-DD), the system can handle multiple target, multiple weapon problems in under two seconds on a Sparc laptop.

Speech Recognition

• Bilmes [2000] applied dynamic Bayesian multinets to speech recognition. Further work in the area can be found at ssli.ee.washington.edu/~bilmes.
• Nefian et al [2002] developed a system for audio-visual speech recognition. This and related research done by Intel Corporation on speech and face recognition can be found at www.intel.com/research/mrl/research/opencv and at www.intel.com/research/mrl/research/avcsr.htm.

Vehicle Control and Malfunction Diagnosis

• Automotive Information Systems (AIS) (www.PartsAmerica.com) developed over 600 Bayesian networks which diagnose 15 common automotive problems for about 10,000 different vehicles. Each network has one hundred or more nodes. Their product, Auto Fix, is built with the DXpress software package available from Knowledge Industries, Inc. (KI).
Auto Fix is the reasoning engine behind the Diagnosis/SmartFix feature available at the www.PartsAmerica.com web site. SmartFix is a free service that AIS provides as an

enticement to its customers. AIS and KI say they have teamed together to solve a number of very interesting problems in order to deliver “industrial strength” Bayesian networks. More details about how this was achieved can be found in the article “Web Deployment Of Bayesian Network Based Vehicle Diagnostics,” which is available through the Society of Automotive Engineers, Inc. Go to www.sae.org/servlets/search and search for paper 2001-01-0603.

• Microsoft Research developed Vista, which is a decision-theoretic system used at NASA Mission Control Center in Houston. The system uses Bayesian networks to interpret live telemetry, and it provides advice on the likelihood of alternative failures of the space shuttle’s propulsion systems.
It also considers time criticality and recommends actions of the highest expected utility. Furthermore, the Vista system employs decision-theoretic methods for controlling the display of information to dynamically identify the most important information to highlight. Information on Vista can be found at research.microsoft.com/research/dtg/horvitz/vista.htm.

• Morjaia et al [1993] developed a system for locomotive diagnostics.

Weather Forecasting

• Kennett et al [2001] learned from data a system which predicts sea breezes.