LEARNING BAYESIAN NETWORKS THROUGH KNOWLEDGE REDUCTION

A Thesis Presented to the Academic Faculty

by

Jorge Pablo Cordero Hernández

In Partial Fulfillment

of the Requirements for the Degree of

Master of Engineering in Computer Engineering

at the

Machine Intelligence Group

Department of Computer Science

Aalborg University

Thesis Committee:

Yifeng ZENG

Anders L. Madsen

Ålborg, Jylland, Denmark

August 2007

Abstract

Learning Bayesian networks from data becomes intractable when a large number of variables are involved in the application domain. Much effort has been made in the past to overcome the computational problem using the divide and conquer strategy. In this Master thesis, it is provided a prior solution to this strategy by introducing a general class of models, named the Bayesian network knots, which explicitly partition the variables into several local components in the network. We propose a learning algorithm called the Overlapping Expansion Learning (OSL) algorithm. Furthermore, we investigate the implications of attribute clustering for learning Bayesian networks. Experimental results show that the OSL is highly competitive. Moreover, we developed a novel attribute clustering algorithm, named the Star Discovery (SD) algorithm. The SD algorithm is able to discover groups of variables with a higher performance than several attribute clustering approaches.

ACKNOWLEDGEMENTS

The author would like to thank Yifeng Zeng for his remarkable supervision and mentoring, Manfred Jaeger for his valuable comments on this work, Uffe Kjaerulff for his support during my Master studies at Aalborg University and the HUGIN company for providing me with the essential software tools.

Contents

Chapter 1

Introduction

A milestone in machine learning, Bayesian networks [1, 2] are employed to represent the probabilistic relationship among random variables. They have been successfully applied in many domains such as the medical, biological, and ecological domains [3, 4]. The core element is to recover a dependency structure in the application, usually called the structural learning of Bayesian networks [5]. Nearly over the past two decades, there has been much research on the problem of learning Bayesian networks from data, which resulted in many effective and efficient learning algorithms [6, 7, 8].

However, learning a complex Bayesian network (composed of a large number of variables) from data is still a difficult task since a huge amount of computation is involved in the learning process. Currently, much effort has been made on this topic, and some desired learning algorithms are appreciated [9, 10, 11, 12, 13]. Most of them adopt the divide and conquer strategy to alleviate the computational problem. They learn a large Bayesian network by recovering small components in the whole network.

For example, the Markov blankets are identified in the sparse candidate algorithm [9] and the max-min hill climbing algorithm [10], the module framework in the learning module networks [11], and the block in the block learning

algorithm [12]. The designing of those algorithms depends on the component formulation in a large network. On the other hand, in general, a component is a local model in a huge network having enormous variables, and represents the reduced expertise or knowledge in the domain.

To draw the reader's attention into the importance to simplify knowledge, consider the following example: Some knowledge engineers or any specialized user may be interested in the specification of the left ulnaris or right ulnaris in the MUNIN Bayesian network [14]; which consists of 1041 variables (on the fourth subset). Figure 1.1 shows the golden MUNIN Bayesian network, notice the agglomeration of variables in different regions. Naturally, subsets of variables in this domain present zones with higher dependencies.

In this work, the local component formulation is researched by introducing the Bayesian network knots. Several methods will be introduced in order to learn and create the knots. We also introduce a new partitioning algorithm denominated the Star Discovery algorithm. The Bayesian network knots cluster some variables into several subsets from a given dataset.

Each Bayesian network knot contains a set of variables having a sound relationship between one another. A Bayesian network knot may be considered a genuine local structure in a large Bayesian network. The Bayesian network knots do not provide only a basic element to recover a large Bayesian network from data, but also present insight into reduced knowledge. Knowledge often overlooked by the global connectivity of a large Bayesian network structure. In this work, several approaches based in data mining (attribute-variable clustering) were formulated to learn the knots from data.

Finally a further BN structure algorithm is introduced. The algorithm is based on the searching in a dependency graph, the identification of sveral local components or knots and then a final combination in order to convey

to the final DAG for a given domain.

The rest of this thesis is organized as follows: Chapter 2 refers to some relevant work and preliminaries. Chapter 3 is an study of different dependency measures. Chapter 4 introduces the notion of dependency graphs and their relevance. Chapter 5 presents several attribute clustering algorithms that are used to perform attribute clustering. In Chapter 6 we introduce the proposed BN learning algorithm. Chapter 7 shows experimental results. Chapter 9 provides a final conclusion on this work and outlines future work.

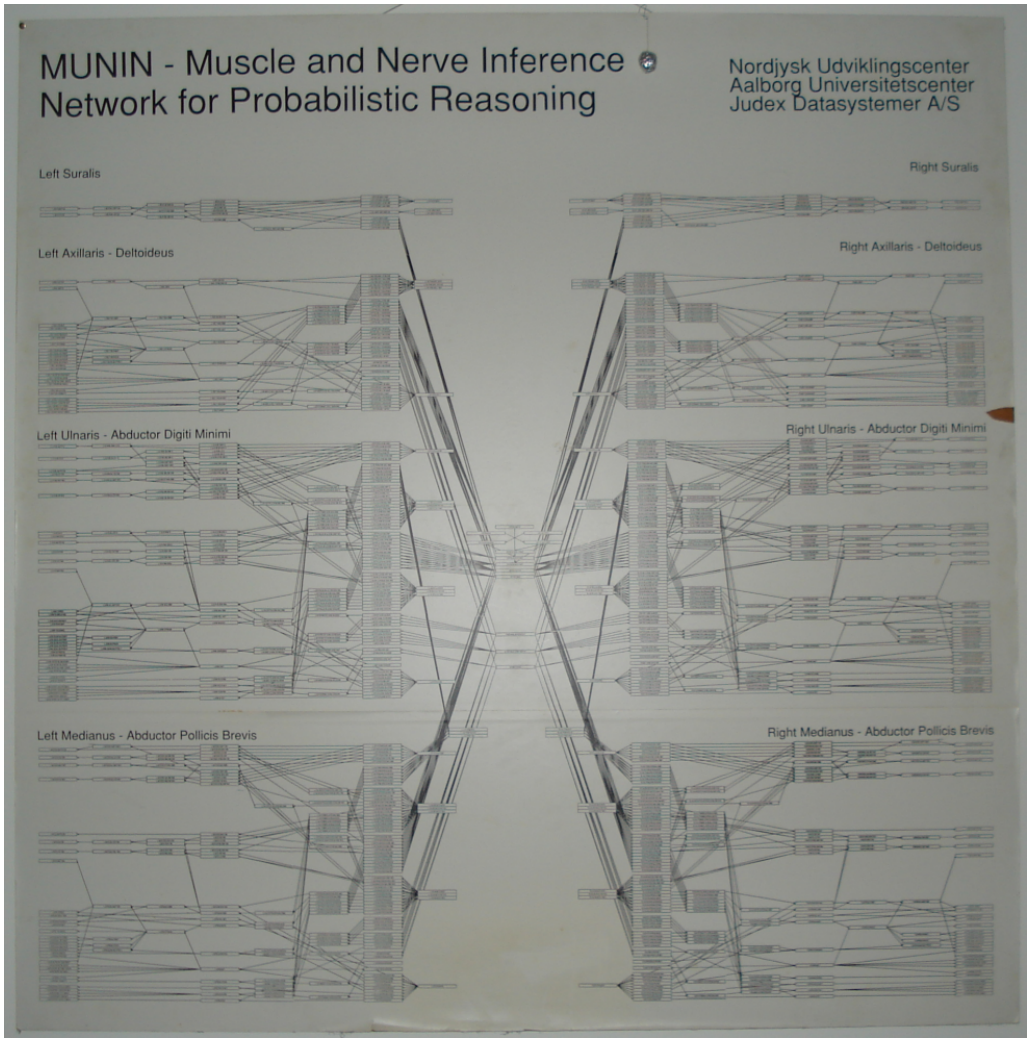


Figure 1.1: The MUNIN Bayesian network.

Chapter 2

Related Work

On this section relevant work is presented, firstly to guide the reader through the necessary background and basic concepts and secondly, it presents an overview of the related work. Mathematical definitions are provided when necessary as well as insightful comments about the relationship between the related work and the current work.

2.1 Bayesian Networks

A Bayesian Network B can be defined as a relation $B = (G, P)$ where $G = (V, E)$ is a directed acyclic graph [15] or DAG (having a set of vertices V representing variables[1] and a set of arcs E which represent causal dependencies between nodes) and P is a probability distribution over G [1, 2].

The set of parents of a variable X_i denoted as $Pa(X_i)$ is the set of nodes in V that have a direct arc pointing to X_i. The set of children of a variable X_i defined as $Ch(X_i)$ is the set of nodes in V that are pointed by X trough an an arc. The set of parents of the children of a variable X_i denoted as

[1]During this work we will use the term variable, vertex and node to refer to attributes or random variables indistinctly. Likewise we can refer to an arc as an edge or also as a link.

$Pc(X_i)$, that is the set of variables in V which point with an arc any element of $Ch(X_i)$. G provides a causal structure in order to establish that a variable X_i is conditionally independent of its non- descendants given its parents $Pa(X_i)$. Therefore the joint probability distribution for the full set of n variables in B is given by:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{2.1}$$

2.1.1 Markov Blanket

Having a DAG $G = (V, E)$ and a variable $X \in V$, the Markov blanket for X stated as $MB(X)$ (being V a set of random variables and E a set of arcs), is the set of variables which make X conditional independent from the rest of the variables $V - MB(X)$. The Markov blanket $MB(X)$ consists of the parents $Pa(X)$, children $Ch(x)$ and the parents of the children $Pc(X)$ of variable X.

2.1.2 Structure Learning in Bayesian Networks

Structure Learning of Bayesian networks is an interesting and challenging task, several approaches have been proposed to perform this task [16, 17, 18, 19, 20, 21, 22, 23, 24]. A general classification for structure learning of Bayesian networks was presented by Kevin Murphy in [25] and has been stated as four different cases:

I. Known structure, full observed data.

II. Known structure, partial observed data.

III. Unknown structure, full observed data.

IV. Unknown structure, partial observed data.

The approach presented in this work can be classified on the third class (refer to Chapter 6 for further details).

Learning Bayesian networks from data is a computationally NP-hard problem and a detailed study on this subject was introduced in [26], and consequently a large amount of work on the field has been dedicated to heuristic-search techniques to identify good models. Two general classes of structural learning algorithms have been widely studied namely the score based methods and the constraint based methods.

Finding a Bayesian network structure with the highest score even from a small sample data with each node having two parents at most is shown to be a hard task. Learning the structure of a Bayesian network is an optimization task, the aim of the score based learning algorithms is to find the structure having the highest statistical score; this process is to exploit a large search space of candidates.

In regard of constraint based learning algorithms, the PC algorithm [6] could be mentioned which is a quite efficient algorithm. The PC algorithm starts with the complete graph over a domain of variables and then it establishes a set of conditional independence statements holding for the data, and uses this set to build a causal network with d-separation properties corresponding to the conditional independence properties.

A good overview of the PC Algorithm was documented in [5] and it was described in detail in [27]] by the following set of steps:

- Step 1 is to begin with a complete graph containing all of the variables.
- In step 2 a variable k is set to zero, whereas k identifies the order of the subset of attributes to be considered for independence tests. Then

for all pairs of nodes X and Y set $DSEP(X, Y) = \emptyset$.

- In step 3 for every adjacent pair of nodes X and Y, remove the arc between them if and only if for all subsets S of order k containing nodes adjacent to X (but not containing Y) the sample partial correlation $rXY.S$ is not significantly different from zero. Add the nodes in S to $DSEP(X, Y)$.

- Step 4 searches tracks the arcs which were removed, it increments k and returns to step 3.

- In step 5, for each triple X, Y, Z in an undirected chain (such that X and Y are connected and Y and Z are connected, but not X and Z), replace the chain with $X \rightarrow Y \leftarrow Z$ if and only if $Y \notin DSEP(X, Z)$.

- Finally step 6 directs back to step 3.

The PC algorithm is probably the widest known, and several improvements and sub developments were devised. Abellan et. al. in [28] proposed a series of variations of the PC algorithm. These variations mainly consist in determining a minimum size of cut sets between pairs of nodes in order to accelerate the process of link deletion and the introduction of a Bayesian score to refine the learned network by a greedy optimization process.

Efforts have been made to improve the structure learning of Bayesian networks. Different approaches for tackling the complexity issue have been adopted. The Sparse Candidate algorithm (SC) [9] is a powerful learning algorithm to recover Bayesian network structures from a large dataset. However it requires two main inputs the network to be sparse and an estimation of the degree of connectivity.

Generally speaking, the SC algorithm takes as input: A dataset $D = \{X_1, X_2, \ldots, X_n\}$, an initial Bayesian network B, a parameter k, and finally

some score function $Score(B|D) = \sum_i Score(X_i|Pa^B(X_i), D)$. The SC algorithm returns a Bayesian network B. For a t number of iterations (until convergence) a restrict step defines a directed graph $H_n = (X, E)$, where $E = \{X_j \rightarrow X_i | \forall_i, X_j \in C_i^n\}$ having that C_i^n ($|C_i^n| \leq k$) is a set of candidate parents for each variable X_i. Afterward, a maximization step finds a Bayesian network $B_n = (G_n, P_n)$ maximizing the $Score(B_n|D)$ among networks that satisfy $G_n \subset H_n$ (i.e., $\forall X_i, Pa^{G_n}(X_i) \subseteq C_i^n$).

By analyzing the SC algorithm it can be sated that the novel idea of the SC is to constrain the search: each variable X is constrained to have parents only from within a predetermined candidate-parents set $C(X)$ of size at most k, where k is defined by the user. Initially, the candidate sets are heuristically estimated, and then hill-climbing (or some other search method) is used to identify a network that (locally) maximizes the score metric. Subsequently, the candidate sets are estimated and another hill-climbing search round is initiated. Cycles of candidate sets estimation and hill climbing are called an iteration. SC iterates until there is no change in the candidate sets or a given number of iterations have passed with no improvement in the network score. The importance of this Sparse candidate algorithm for the present work relies in that our proposed algorithm could fit in the class of SC algorithms. Thus, it is an hybrid and modular approach thought to overtake the complexity in several domains.

SC and several algorithms like Max-Min Hill-Climbing (MMHC) [29] try to speed up the search process in the search space candidates. Some researchers have used data mining techniques such as Evolutionary programming [30] to learn the Bayesian networks. In this approach in the conditional phase, dependency analysis is conducted to reduce the size of the search space. In the search phase, the good Bayesian network models are generated

by using an evolutionary algorithm.

Feature selection [31] also has been used in order to learn the Bayesian networks more efficiently, by selecting the most indicating values to generate networks which are computationally simpler to evaluate. Several approaches have been adopted to break the Bayesian networks into smaller building blocks and then to learn these blocks separately which will be less costly regarding the computational costs and finally to aggregate these blocks to recover the full network, see [12]. Some researchers have proposed the idea of module networks where it could be used in domains like stock market or biotechnology where many variables could have similar behaviors. Hence, this method tries to partition the data based on the variables sharing the same conditional probability distribution and encapsulate a set of such similar variables formally in a structure called modules [11].

However, research work has not been directed on the study of identifying truly subsets of Bayesian network structures. Centralized structure learning of Bayesian networks is by no means the only concern in knowledge engineering. As a matter of fact the vast number of databases in the world has a distributed character.

Computing the Bayesian network over distributed data can be a difficult task. Nevertheless, whenever the data is separated in several locations and it is heterogeneous, the structure of a Bayesian network can be learned by applying the collective learning algorithm designed by Sivakumar et al. in [32].

In the later algorithm the Bayesian network can be acquired with the same precision as if the data were centralized into a single place. Thus, the collective learning algorithm consists of four main steps: local learning, sample selection, cross learning and combination. This collective learning

algorithm (specially the last step) is interesting for the current work that is to be presented in this thesis. Thus, the arrangement of subsets of variables can be seen as a set of distributed variables across many locations then a later step should unite the subsets to complete the full Bayesian network. Another framework to learn Bayesian networks from distributed data was proposed by Li et al. [33].

Interestingly enough, it is clear that much work has been done in the past for distributed structure learning of Bayesian network, but still a good motivation in this work is to introduce the notion of local components. These local components can also be used for other applications such as gene selection in gene expression data or knowledge simplification over massive domains. However, we will adhere to the orthodox paradigm of learning BN from a single source.

2.1.3 Learning the Structure of BN from Small Datasets

Real life data such as those ones in the biomedical databases are relatively small when compared to their number of variables. Logically, domains such as research in genes and genomes produce a huge number of variables and a considerably small set of samples. Thus, depending of the area of study, samples can be subtracted only from a small group of subjects. A dramatically true example on this assertion is the colon cancer dataset [34] which has 2000 variables and 62 samples and the leukemia dataset [35] has only 72 samples and 7 129 variables. The small sample size issue is not exclusive for medical applications but it may affect any domain in which a dataset is small enough to produce a non realistic model.

Evidently, working with small and reduced datasets is a major issue when referring to practical purposes. Modeling a domain with a reduced observa-

tion and mapping it to a Bayesian Network is a challenging task. Two fundamental concerns are involved in this case: Learning the BN structure and learning the parameters from data. Even though, a general rule for deciding the sample size for accurate learning of Bayesian networks does not exist, empirical research have provided a particular convergence point in which the structure and parameters do not change with a very high amount.

Accurate structural learning of Bayesian networks depends mainly on the domain in question. Thus, for some datasets the Bayesian network can be rapidly found; for others may need an extensive observation in order to return a comprehensive and reliable DAG.

In terms of structure learning of Bayesian networks (having a reduced sample size), Carrillo et al. proposed in [36] a score based algorithm in which the structure learning task is modeled as an optimization problem. By using simulated annealing as the search procedure and a given Bayesian score as a measure of goodness, the selection of parents of a given node is restricted. Finally a pruning phase of the algorithm eliminates incorrect edges. In a slightly different field, Agnieszka et al. [37], developed a new method for learning parameters from small datasets was presented and tested with good results using a rather small but not trivial Bayesian Network, the HEPAR Network [38]. The novel idea of the later work was to present a noisy OR gate in order to reduce the amount of data necessary to complete several conditional independence tests. The denominated "noisy OR" has the capability to approximate some conditional probability distributions. Thus, an elementary binary noisy OR can be sated as:

"*If there exists a variable Y having a set of parents $X = Pa(Y)$ and each variable in X has a mutually exclusive probability pi of affecting Y the*

following expression.".

$$pi = Pr(y|\bar{x_1}, \bar{x_2}, \ldots, barx_i, \ldots, \bar{x_{n-1}}, \bar{x}) \tag{2.2}$$

can be reformulated to:

$$pi = Pr(y|\bar{x_1}, \bar{x_2}, \ldots, barx_i, \ldots, \bar{x_{n-1}}, \bar{x}) \tag{2.3}$$

whereas $\bar{x_1}, \bar{x_2}, \ldots, barx_i$ denotes the absence of the causes x_1, x_2 and the presence of x_i.

The mutually exclusive assumption aided to catch an artificial independence test. The leaky noisy OR is a more sophisticated idea to apply the same principle to categorical variables. With no doubt this is a true example of how an "external mechanism" can bypass the conventional calculation of probability distributions. In essence a similar idea is introduced in the present work for structure learning, an external process (an information approach) can detect sets of correlated variables which can be immediately considered for structural learning in small groups.

2.1.4 Attribute Clustering

Attribute clustering has been previously employed to detect the inner dependence between subsets of variables, especially in genetics due to its complex nature (both on small and sufficient sample sizes). Several conventional clustering algorithms have been applied to re-group and reveal subsets of correlated genes such as: K-means algorithm, fuzzy clustering, self organizing maps, hierarchical clustering and gene selection techniques.

An accurate information based method for this task is the k-modes algorithm and it was presented by Au et. al. in [39]. A set of highly correlated

variables is obtained by computing an information theoretic measure that is properly introduced in Chapter 3. Briefly speaking, the k-modes algorithm works on the following way: First it is initialized with k clusters, each of them having one mode attribute (a mode is an attribute with the highest dependency among variables). The second step assigns an attribute X_j to a cluster C_i with the highest proximity. The third step computes a new mode for each cluster by finding the new best local correlated variable. Finally the process is repeated for a number of iterations or if the modes in each clusters do not change.

What was the reason to relate attribute clustering to this research? Despite of the fact that the work presented on this report does not perform attribute clustering, it is believed that future development can yield to a comprehensive unified mechanism for attribute clustering. Therefore, the main algorithm or variations from it would perform attribute clustering besides of structural learning and knowledge reduction over the knots. The reason to state this is the confidence on the k-modes method to produce truly correlated sets of variables. Actually the k-modes approach was applied to a discrete set of variables. Obviously, that method can be applied to any domain since it is based on an information metric and not in a very specific metric as distance measurement or Pearson's correlation. A more detailed explanation on future work on this matter is presented in chapter 6.

Chapter 3

A Universal Dependency Estimator

A major objective of machine learning is to gain knowledge domain. This task often regarded as learning by experience aims to acquire certain understanding of a domain given (continuous or discrete) Data. As seen in chapter 1, a domain U (for the effect of the present work) can be described by a set X of random variables having Ω observations.

An initial step before detecting knowledge is to find a correlation between variables. This correlation estimate has to serve as a dependency metric. This chapter presents a comprehensive study of several meaningful metrics to test relationships between those variables. Finally, I present the normalized mutual information; an all-purpose field-independent dependency metric.

Euclidean distance and Pearson correlation have been previously used to find meaningful relationships between genes [40, 41]. Nevertheless, the results of these estimates are absolute magnitudes; thus, difficult to generalize for other domains. They are useful if all variables $\{X_1, \ldots, X_n\} \in X$ can take values from equal intervals (implying the same cardinality when working with discrete variables). Hence, they incorporate inaccurate results whenever the range (cardinality) of the random variables is sparse. Consider a simple

example inspired in terms of Euclidean distance:

Example 3.1. *Consider two pairs of points* $p1, p2$*, and* $p3, p4$ *in a two dimensional Euclidean space (defined by a X, Y axis).* $p1$ *and* $p2$ *are situated in the coordinates* $(1000, 1000)$ *and* $(500, 500)$ *respectively.* $p3$ *is in the point* $(10, 10)$ *and* $p4$ *is in* $(9, 9)$*. The Euclidean distance between two points situated in the coordinates* $(x1, y1)$ *and* $(x2, y2)$ *is:*

$$d(p1, p2) = d((x1, y1), (x2, y2)) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \tag{3.1}$$

By using Equation 3.1 we obtain that the Euclidean distance for the first pair is $d(p1, p2) = \sqrt{500000}$ *and* $d(p3, p4) = \sqrt{2}$*. Therefore,* $d(p1, p2) >> d(p3, p4)$*. Now, suppose we impose a limit in the range of* $p1$ *and* $p2$ *to take values from 0 to 1000 in the domain and counter domain; then, we restrict the range of* $p3$ *and* $p4$ *to take values from 0 to 10. However, the later comparison is not fair since* $p3$ *and* $p4$ *were limited to a smaller space whatsoever.*

Instead, if we divide each of the terms $(x1 - x2), (y1 - y2)$ *by 1000 for* $p1, p2$*, and by 10 for* $p3, p4$*; then, the relative proportions are found and the comparison of distances is admissible.*

We can adapt Equation 3.1 to test dependency between two continuous random variables X_i and X_j. These variables take the values $w_{i,k}, w_{j,k} \in \Re$ for $k = 1, \ldots, \Omega$ samples. Then, the Euclidean distance between two random variables can be defined as:

$$d(X_i, X_j) = (\sum_{k=1}^{\Omega} (w_{i,k} - w_{j,k})^2)^{\frac{1}{2}} \tag{3.2}$$

Holding the same notation from the previous formula, a more robust

metric named the Pearson correlation coefficient [39] is defined as:

$$PC(X_i, X_j) = \frac{\sum_{k=1}^{\Omega}(w_{i,k} - \bar{w_i})(w_{j,k} - \bar{w_j})}{(\sum_{k=1}^{\Omega}(w_{i,k} - \bar{w_i})^2)^{\frac{1}{2}}(\sum_{k=1}^{\Omega}(w_{j,k} - \bar{w_j})^2)^{\frac{1}{2}}}, \tag{3.3}$$

whereas $\bar{w_i}$ and $\bar{w_j}$ are the mean values $\frac{1}{\Omega}\sum_{k=1}^{\Omega} w_{i,k}$, $\frac{1}{\Omega}\sum_{k=1}^{\Omega} w_{j,k}$ for X_i and X_j respectively. This coefficient measures the strength of the linear relationship between the random variables. However, the application of the previous distance-based metrics into discrete and complex domains is not clear. Hence, there is the need to device a universal metric which could be applied to either continuous or discrete domains; moreover, it has to be independent of the domain in study.

3.1 Information Theory Based Approaches

I continued the search for a true proximity measure that could really express certain level of dependency. Information theory metrics such as entropy (firstly introduced by the prominent American mathematician Claude E. Shanon in 1948 [42]) and mutual information rely in the concept of uncertainty among variables.

These *information-based distances* were founded by Shannon in the field of communication theory. Paraphrasing Cover et al. in [43]: *"...entropy is the minimum descriptive complexity of a random variable, and mutual information is the communication rate in the presence of noise."* Therefore, we define dependency as the lack of uncertainty among a set of variables.

3.1.1 Information Entropy

The (marginal) entropy of a single discrete random variable X_i having a set of possible states $X_i = \{x_1^i, \ldots, x_l^i\}$ and cardinality l is given by:

$$H(X_i) = \sum_{k=1}^{l} p(x_k^i) \log_2 p(x_k^i), \tag{3.4}$$

where $p(x_k^i)$ is the marginal probability distribution that variable X_i is instantiated to state k. Conclusively, as stated in [43], the entropy of a random variable is its average uncertainty; it is minimal in the extreme values 0 and 1. Figure 3.1 depicts an example of a Bernoulli trial. The behavior of the marginal entropy for X_i always increases as the probabilities tend to be equally distributed among the possible outcomes (instantiations), such that if $\forall_{(x_a^i, x_b^i \in X_i, a \neq b)} x_a^i = x_b^i$ then $H(X_i)$ is maximal.

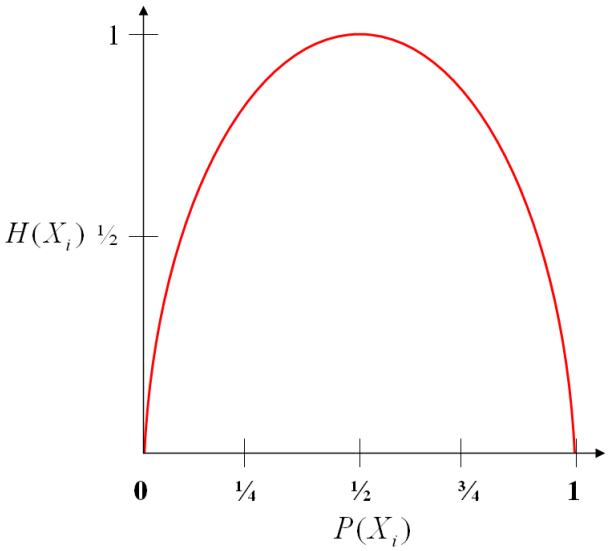


Figure 3.1: The Entropy of a Bernoulli trial. Two outcomes are equally possible. Therefore, $H(X_i) = 1$ iff $P(X_i) = 0.5$.

3.1.2 Joint Entropy

The joint entropy between two random variables X_i and X_j is given by the following formula:

$$H(X_i, X_j) = \sum_{a=1}^{l} \sum_{b=1}^{m} p(x_a^i, x_b^j) \log_2 p(x_a^i, x_b^j), \tag{3.5}$$

where the cardinality of X_i and X_j is l and m respectively. Moreover, we can extend and generalize Equation 3.5 for more than two random variables:

$$H(X_1, \ldots, X_n) = \sum_{a=1}^{l}, \ldots, \sum_{z=1}^{m} p(x_a^1, , \ldots, x_z^n) \log_2 p(x_a^1, \ldots, x_z^n) \tag{3.6}$$

Remark 3.1. *The estimation of probability distributions is reduced to the counting of frequencies. Assume σ is the selection operator [44] and R is a relation. Table 3.1 shows the methodology used in this work to estimate probabilities among discrete random variables. I use a notation similar to [39] to describe the counts. Consider $|\sigma^*(R)|$ as the select-all operation ($|\sigma_{X_i \neq \emptyset}(R)|$ for an arbitrary variable X_i) returning the Ω total of samples in the Data D.*

Table 3.1: Frequency Estimation.

Probability Distributions	Counts
$p(X_i = x_k^i)$	$\frac{\lvert\sigma_{X_i = x_k^i}(R)\rvert}{\lvert\sigma^*(R)\rvert}$
$p(X_i = x_a^i \wedge X_j = x_b^j)$	$\frac{\lvert\sigma_{X_i = x_a^i \wedge X_j = x_b^j}(R)\rvert}{\lvert\sigma^*(R)\rvert}$

Analogously to the discrete versions of entropy, suppose $p(x_i, x_j)$ is the joint probability distribution function for X_i and X_j, and $p(x_i)$ is the marginal probability distribution function for X_i; then, Equations 3.4, 3.5 and 3.6 are

rewritten to meet continuous domains:

$$H(X_i) = \int_{x_i \in X_i} p(x_i) \log_2 p(x_i d(x_i)) \tag{3.7}$$

$$H(X_i, X_j) = \int_{x_i \in X_i, x_j \in X_j} p(x_i, x_j) \log_2 p(x_i, x_j) d(x_i) d(x_j) \tag{3.8}$$

$$H(X_1, \ldots, X_n) = \int_{x_i \in X_i, \ldots, x_n \in X_n} p(x_1, \ldots, x_n) \log_2 p(x_1, \ldots, x_n) d(x_i), \ldots, d(x_j) \tag{3.9}$$

Remark 3.2. *Any continuous random variable $X_p \in X$ with a probability density function $f(X_p)$ can be discretized by bounding its domain into Z intervals.*

3.2 Algorithmic Information Theory

Andrey Kolmogorov, a brilliant Russian mathematician extended the notion of probabilistic information entropy into a more general concept, the descriptive complexity. Instead of representing X_i as a random variable, all of its observations are compressed into a binary code. The aim is to find the shortest description $d(X_i)$ of a program p that can output X_i by using a universal computer UC. Thus, the descriptive complexity is defined as $K_{UC}(X_i) = argmin_{p:UC(p)=x}|d(X_i)|$ or just $K(X_i) = |d(X_i)|$.

Similarly, $K(X_i X_j)$ is formulated as the complexity of encoding the concatenation of X_i and X_j. $K(X_i|X_j)$ is the conditional algorithmic complexity between X_i and X_j and can be interpreted as the minimal description necessary to output X_i receiving X_j as side information [45]. The following

equivalence was proved in [45]:

$$0 \leq K(X_i|X_j) \approx K(X_iX_j) - K(X_j) \leq K(X_i) \tag{3.10}$$

A complete analysis of Kolmogorov complexity can be found in [43]. A useful tool called the Normalized compression distance (NCD) to compute algorithmic information estimates was presented in [45]. Kraskow et al. [46] adapted the NCD to Shannon's probabilistic entropy.

3.3 Mutual Information

The mutual information, transinformation or "information" $I(X_i, X_j)$ is a "conceptual distance" between two random variables, namely X_i and X_j [42]. This information tool provides useful estimates regarding dependency or interaction between pair-wise relationships among two or more random variables[1].

Indeed, $I(X_i, X_j)$ is the average reduction in uncertainty of variable X_i by knowing X_j [43]. The higher the mutual information, the better X_i can be predicted due to the knowledge of X_j and vice versa. Conclusively, the mutual information can be used either to test the degree of relationship between discrete or continuous random variables. The mutual information between two discrete random variables $X_i, X_j \in X$ (holding the notation of Equation 3.5) can be defined as:

$$I(X_i, X_j) = \sum_{a=1}^{l} \sum_{b=1}^{m} p(x_a^i, x_b^j) \log_2 \frac{p(x_a^i, x_b^j)}{p(x_a^i)p(x_b^j)} \tag{3.11}$$

[1] Another notation which is frequently used to describe mutual information in literature is $I(X_i; X_j)$.

On the other hand if X_i and X_j are continuous, then the mutual information is:

$$I(X_i, X_j) = \int_{x_i \in X_i, x_j \in X_j} f(x_i, x_j) \log_2 \frac{f(x_i, x_j)}{f(x_i) f(x_j)} d(x_i) d(x_j), \tag{3.12}$$

whereas $f(x_i, x_j)$ is the joint density for both variables; $f(x_i)$ and $f(x_j)$ are the densities for X_i and X_j respectively. In the most general case for any arbitrary pair of random variables, the mutual information is identical to the least upper bound of computing Equation 3.11 for the quantization of X_i and X_j in P and Q partitions respectively [43], Equation 3.13 presents this generalization.

$$I(X_i, X_j) = sup_{P,Q} I([X_i]_P, [X_j]_Q), \tag{3.13}$$

where $[X_i]_P$ is the quantization (discretization) of X_i in a collection P of disjoint sets $\bigcup_i P_i = \chi_i$ (χ_i is the range of X_i). Similarly, $[X_j]_Q$ follows the previous definition for X_j. In all cases 3.13 monotonically increases iff $|P|$ or $|Q|$ increase.

Remark 3.3. *The base of the logarithm in Equations 3.11 and 3.12 specifies the units in which the mutual information is measured. A base 2 logarithm measures $I(X_i, X_j)$ in bits, whereas a base* exp *logarithm measures the mutual information in nats. Generally, a base of 2 is assumed in literature; for the effect of this work logarithms base 2 are employed (unless specified).*

If we wish to denote information metrics using another base for the logarithm we write the base b as a subindex next to the literal of the function in question. For example if $X_1, X_2 and X_3$ are random variables then $H_b(X_1), H_b(X_1, X_2)$ and $I_b(X_1, X_2)$ are the entropy, joint entropy and mutual information functions using a base b.

3.3.1 Properties and important remarks regarding mutual information

The mutual information is simetric $I(X_i, X_j) = I(X_j, X_i)$ and non-negative $I(X_i, X_j) \geq 0$. Actually, a value of $I(X_i, X_j) = 0$ represents strict independence between the random variables X_i and X_j such that $p(X_i, X_j) = p(X_i)p(X_j)$ holds. A fundamental remark is the equivalence between mutual information and Shannon entropy. The following properties describe several relationships:

$$I(X_i, X_j) = H(X_i) - H(X_i|X_j) \tag{3.14}$$

$$I(X_i, X_j) = H(X_j) - H(X_j|X_i) \tag{3.15}$$

$$I(X_i, X_j) = H(X_i) + H(X_j) - H(X_i, X_j) \tag{3.16}$$

$$I(X_i, X_i) = H(X_i) \tag{3.17}$$

Remark 3.4. *Equation 3.18 is the marginalization of x_a upon x_b (l is the cardinality of the set of instantiations $\{x_1, x_2, \ldots, x_l\}$):*

$$p(x_a) = \sum_{b=1}^{l} p(x_a, x_b) \tag{3.18}$$

In the case of Equations 3.14 and 3.15, these quantities are substituted and proved to be equivalent to mutual information [43] by the following:

$$I(X_i, X_j) = \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) \log_2 \frac{p(x_a^i|x_b^j)}{p(x_a^i)} = \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) \log_2 \frac{p(x_b^j|x_a^i)}{p(x_b^j)} \tag{3.19}$$

Now, by substituting in Equation 3.14:

$$= -\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 p(x_a^i) + \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 p(x_a^i|x_b^j)$$

$$= -\sum_{a=1}^{l} p(x_a^i)\log_2 p(x_a^i) - \left(-\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 p(x_a^i|x_b^j)\right)$$

$$= H(X_i) - H(X_i|X_j)$$

Equivalently, Equation 3.15 becomes:

$$= -\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 p(x_b^j) + \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 p(x_b^j|x_a^i)$$

$$= -\sum_{b=1}^{m} p(x_b^j)\log_2 p(x_b^j) - \left(-\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 p(x_b^j|x_a^i)\right)$$

$$= H(X_j) - H(X_j|X_i)$$

The equality stated in 3.16 is a well known rule for mutual information. It provides an idea regarding the level of knowledge or dependency between two random variables (the remaining information which results from the difference of the sole average uncertainties minus the joint average uncertainty). Obviously, mutual information is symmetric since 3.16 is always symmetric:

$$I(X_i, X_j) = \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)\log_2 \frac{p(x_a^i, x_b^j)}{p(x_a^i)p(x_b^j)}$$

$$= \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) log_2 p(x_a^i, x_b^j) - \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) log_2 p(x_a^i) p(x_b^j)$$

$$= -\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j)(log_2 p(x_a^i) + log_2 p(x_b^j)) - \left(-\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) log_2 p(x_a^i, x_b^j) \right)$$

$$= -\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) \log_2 p(x_a^i) - \sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) \log_2 p(x_b^j) - \left(-\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) log_2 p(x_a^i, x_b^j) \right)$$

$$= -\sum_{a=1}^{l} p(x_a^i) \log_2 p(x_a^i) - \sum_{b=1}^{l} p(x_b^j) \log_2 p(x_b^j) - \left(-\sum_{a=1}^{l}\sum_{b=1}^{m} p(x_a^i, x_b^j) log_2 p(x_a^i, x_b^j) \right)$$

$$= H(X_i) + H(X_j) - H(X_i, X_j)$$

The equivalence in 3.17 is also denominated *self information* and can be proved by showing a similar substitution as in 3.14:

$$I(X_i, X_i) = H(X_i) - H(X_i|X_i)$$

$$p(x_a^i, x_a^i) = 0, p(x_a^i|x_a^i) = \frac{p(x_a^i, x_a^i)}{p(x_a^i)} = 0$$

$$= -\sum_{a=1}^{l} p(x_a^i) \log_2 p(x_a^i) - \left(-\sum_{a=1}^{l}\sum_{a=1}^{l} p(x_a^i, x_a^i) \log_2 p(x_a^i|x_a^i) \right)$$

$$= -\sum_{a=1}^{l} p(x_a^i) \log_2 p(x_a^i) - 0$$

$$= H(X_i)$$

A final equivalence is the connection between mutual information and relative entropy:

$$I(X_i, X_j) = D(p(x_a^i, x_b^j)||p(x_a^i)p(x_b^j)) = \sum_{a=1}^{l}\sum_{b=1}^{m}(p(x_a^i, x_b^j)) \log_2 \frac{(p(x_a^i, x_b^j))}{(p(x_a^i)p(x_b^j))} \tag{3.20}$$

For a complete description of the proofs from the previous properties refer to [43].

3.3.2 Chain Rule for mutual information

Mutual information can be used to calculate the dependency level for more than 2 random variables. Intuitively, Equation 3.14 can be extended to admit a set $S \subseteq X$ of variables $((|S| = r) \wedge (r > 2))$ instead of two. Therefore, the information for r variables is given by the recursive sum of the conditional entropies over subsets of S. Assume that $S = \{X_1, X_2, \ldots, X_r\}$ is the set of random variables, an arbitrary variable $X_q \in S$, and $X_h = S \backslash X_q$ such that $|X_h| = o$.

$$I(X_1, X_2, \ldots, X_o, X_q) = \sum_{i=1}^{o} I(X_i, X_q | X_{i-1}, X_{i2}, \ldots, X_1) \tag{3.21}$$

The later Equation is found after using identity 3.21 in X_h and X_q:

$$I(X_h, X_q) = H(X_h) - H(X_h, X_q)$$

$$= H(X_1, X_2, \ldots, X_o) - H(X_1, X_2, \ldots, X_o|X_q)$$

$$= \sum_{i=1}^{o} H(X_i|X_{i-1}, \ldots, X_1) - \sum_{i=1}^{o} H(X_i|X_{i-1}, \ldots, X_1, X_q)$$

$$= \sum_{i=1}^{o} I(X_i, X_q|X_{i-1}, X_{i2}, \ldots, X_1)$$

Two key definitions are necessary to substitute the right hand side of 3.21. The first one is the expansion rule for entropy between 2 or more variables:

$$H(X_i, X_j) = H(X_i) + H(X_j|X_i) \tag{3.22}$$

Consequently, 3.22 is used to generalize the second requirement, the joint entropy for o variables $\{X_1, X_2, \ldots, X_o\}$:

$$H(X_1, X_2, \ldots, X_o) = \sum_{i=1}^{o} H(X_i|X_{i-1}, \ldots, X_1) \tag{3.23}$$

$$= H(X_1) + H(X_2|X_1) + \ldots + H(X_k|X_{k-1}, X_{k-2}, \ldots, X_1) + \ldots$$

$$+H(X_o|X_{o-1}, X_{o-2}, \ldots, X_1)$$

Equation 3.23 is also called the chain rule for entropy and it is equivalent to 3.6 but easier to compute since the chain rule factorizes the extensive joint entropy into several terms. Two alternate definitions of the chain rule have also been stated: Total correlation [47, 48] and interaction information [49, 50]. These approaches can be understood as generalizations of mutual

information.

3.3.3 Total Correlation

The total correlation $C(X_1, X_2, \ldots, X_s)$ between s variables is the difference between the marginal entropies (assuming that all of the variables are independent of each other) and the joint entropy of all variables:

$$C(X_1, X_2, \ldots, X_s) = \sum_{i=1}^{s} H(X_i) - H(X_1, X_2, \ldots, X_s) \tag{3.24}$$

Actually, it is easy to device that total correlation is a multi characterization of Equation 3.16 (trying to generalize in a sense from 2 to s random variables). The motivation of total correlation is to provide the simplest measure of relationship between s random variables based in the most general (loosest) estimation. Therefore, total correlation disregards any other possible calculation of entropy among other subsets of variables. Total correlation is symmetric and non zero since $H(X_i) \geq H(X_1, X_2, \ldots, X_s)$ (a value of 0 indicates total independence between the s random variables).

The main inconvenient of total correlation is that it might not provide a realistic estimate of dependency between n variables. The marginal average uncertainties and the largest joint uncertainty may not have been enough to capture the real dependency among other (possibly important) subsets of variables.On the other hand, the total correlation explores only a subset of the arrangements of variables considered in the initial chain rule used to calculate the mutual information. Therefore, the computation of the total correlation is faster than the one in the chain rule.

3.3.4 Interaction Information

Another interesting tool for testing the level of stochastic dependency between a collection of variables is interaction information [50, 51]. Interaction information calculates the "synergy" between a $k-way$ arrangement of variables ($|X| = k$). Jakulin et al. [51] made a distinction between the semantics of dependency and interaction. An interaction is an atomic relationship between a pair of variables only. Dependency can be established between two or more variables.

The interaction information for a set of variables $X = \{X_1, X_2, , X_n\}$ is defined by the following formula:

$$II(X) = -\sum_{S \subseteq X} (-1)^{|X|-|S|} H(S), \tag{3.25}$$

whereas, $H(S)$ is the entropy over a set $S \subseteq X$ of variables. Thus, the $k-way$ interaction information can be interpreted as the sum of the joint entropies of the power set $2^{|S|}$ of S (omitting the empty set). Interaction information can also be expressed in terms of conditional entropy. Let $X_i \in X$, then the $k-way$ interaction information is given by:

$$II(X, X_i) = II(X|X_i) - II(X) \tag{3.26}$$

The main drawback of interaction information is its (vague) interpretation. In the simplest case the mutual information is obtained by substituting the two variables in Equation 3.25; logically, the properties of mutual information (non zero and symmetry) hold. Regardless of the number of random variables, interaction information is always symmetric. However, if $|X| \geq 3$, then positive and negative values are possible.

A positive interaction information indicates that the variables in X contain a higher dependency level (synergy) than in S (Having that $S \subset X$ and $|S| = |X| - 1$). In contrast, a negative value of interaction information denotes redundancy among the elements of X. Thus, there exist $|X| - 1$ values contained in X sharing a higher relationship. The only admissible independence test for this estimate is: $II(X_1, X_2, \ldots, X_n) = -II(X_1, X_2, \ldots, X_{n-1})$ iff the variables in the set $X_1, X_2, \ldots, X_{n-1}$ become independent given X_n.

The next section presents the differences between the chain rule of mutual information, total correlation and interaction information.

3.4 Towards an efficient computation of information

The calculation of mutual information between 2 or more random variables, total correlation and interaction information rely on the calculation of several entropies. In any case the computation is not simple due to the estimation of joint probability distributions (a frequency counting problem).

Several attempts have been made in the past to approximate the calculation of information. In the case of mutual information Vilmansen [52] establishes that it is possible to approximate mutual information if we assume some prior probability distributions. The main problem is to find a function which efficiently approximates joint probability distributions.

Ultimately, the objective is to reduce the time complexity of the counting of frequencies. Friedman et al. stated in [53] that the time required in order to estimate frequencies can be optimized by reducing the sample size of the domain. This mechanism reduces the complexity of the counting problem by a factor $\beta < 1$. This threshold $\beta * \Omega$ defines a point in which the mu-

tual information, entropy or the target function converges in an acceptance interval (reaching sufficient statistics). Thus, the function in question does not significantly improve its value by increasing the sample size. However, the selection of β is subject to the complexity of the domain in study.

Polynomial density expansions based on cumulants [54] have been previously used to approximate the joint probability functions. Another compacting simplification of joint probabilities is acquired by defining a latent attribute [55] with a lower cardinality than the ones in the original variables. A more refined approach is the one proposed by Hyvärinen in [56]. This method is a robust approximation of marginal entropies based on the concept of maximum entropy density. Specifically, this approach approximates the entropy $H(X_i)$ of a random variable X_i with a density function $\Delta(x_i)$.

$$H(X_i) = -\int \Delta(x^i) log \Delta(x^i) d(x^i) \tag{3.27}$$

Assuming that the density function $\Delta(x^i)$ is quantisized according to a collection of Gaussian functions into m intervals $C = \{c_1, c_2, \ldots, c_m\}$:

$$\int \Delta(x^i) G_j d(x^i) = c_j, j = 1, \ldots, m \tag{3.28}$$

Then, the marginal entropy can be approximated as:

$$H(X_i) \approx -\int \Delta(x^i) log \Delta(x^i) d(x^i) \approx H(V^i) - \frac{1}{2} \sum_{j=1}^{m} c_j^2, \tag{3.29}$$

where $H(V^i)$ is the entropy of a standardized Gaussian variable and it is equal to $\frac{1}{2} log 2\pi$ [56]. This method offers good marginal approximations of entropy. Nevertheless, the question still remains open for joint entropy estimations. The study of a concrete and efficient approximation of multi-

variate probability distributions is a theme which is beyond the scope of this work. Moreover, a comparison between approximate and true estimates of information is regarded as future work. Further details are provided in chapter FUTUREWORK. In this work for practical purposes; I used the data structures described in chapter XXX to efficiently compute the queries when necessary.

3.4.1 Normalized Mutual Information

Mutual information is a sound estimate to describe dependency. Nevertheless, its analysis as a true distance metric is decisive since we want to device a suitable dependency estimator. As previously mentioned in [39, 46], mutual information is probably the best option to define a degree of dependency or interaction between two or more variables. However, it is obvious to appreciate that its definition as formulated in Equation 3.11 is biased by the cardinality of the variables involved in the calculation of the marginal and joint probability distributions.

Indeed, mutual information can be regarded as an absolute metric; thus, it is not completely accurate to reflex true dependencies in a realistic domain. When working in massive domains, it is unlikely that all attributes would have the same cardinality. Therefore, there is the need to find a metric which is relative depending on the cardinality of a given set of variables. Li et al. introduced in [45] a normalized universal distance (NID) using algorithmic mutual information. The NID as any other proper metric is based upon the following definition.

Definition 3.1. *Let X be a set of variables and $X_i, X_j, X_k \in X$. A distance $D(X_i, X_j)$ between X_i and X_j is a true metric iff $D(X_i, X_j)$ holds the **identity axiom**: $D(X_i, X_j) = 0$ iff $X_i = X_j$;*

$D(X_i, X_j)$ *holds the* ***symmetry axiom****:* $D(X_i, X_j) = D(X_j, X_i)$*;*

$D(X_i, X_j)$ *holds the* ***triangle inequality axiom****:* $D(X_i, X_j)+D(X_j, X_k) \geq D(X_i, X_k)$.

Finally, the NID between two sequences S_i and S_j consist of the following expression:

$$NID(S_i, S_j) = \frac{K(S_i|S_j) + K(S_j|S_i)}{K(S_i, S_j)}, \tag{3.30}$$

or simply

$$NID(S_i, S_j) = 1 - \frac{I_{alg}(S_i|S_j)}{K(S_i, S_j)}, \tag{3.31}$$

whereas $I_{alg}(S_i, S_j) = K(X_j) - K(X_j|X_i)$ is the algorithmic mutual information. Recall that $K()$, $K(.,.)$ and $K(.|.)$ are the marginal, joint and conditional Kolmogorov complexities (Section KOLMOGOROV). However, they showed that the previous estimate might produce not accurate results if we use more than 2 variables. This argument made them device the following metric:

$$NID'(S_i, S_j) = \frac{argmax\{K(S_i|S_j), K(S_j|S_i)\}}{argmax\{K(S_i), K(S_j)\}} \tag{3.32}$$

Furthermore, the two NID variants have been adapted from its Kolmogorov version into the terms of Shannon's entropy by Kraskov et al. in [46]. Therefore, the mutual information distances $D(X_i, X_j)$ and $D'(X_i, X_j)$ between two random variables X_i and X_j are defined as:

$$dI(X_i, X_j) = 1 - \frac{I(X_i, X_j)}{H(X_i, X_j)} \tag{3.33}$$

$$dI'(X_i, X_j) = \frac{argmax\{H(X_i|X_j), H(X_j|X_i)\}}{argmax\{H(X_i), H(X_j)\}} \tag{3.34}$$

Both of these formulas are not identical. Nevertheless, they have been

empirically proved to be equivalent with some minimal deviation between each other (refer to [46] for further details). Another measure was employed by Au et al. [39] in the ACA algorithm as the central metric in order to test dependency between pairs of random variables. The ACA algorithm uses a similar version of Equation 3.33:

$$R(X_i, X_j) = \frac{I(X_i, X_j)}{H(X_i, X_j)} \tag{3.35}$$

This estimate was firstly shown in [ACAAAAAA56] in the bioinformatic's field, it is called the interdependency redundancy measure or i.r.m. and fullfils the identity, symmetry and triangle inequality axioms. The interdependence redundancy measure is a straightforward normalization of mutual information and mirrors Equation 3.33. A low value for Equation 3.33 represents a short distance (high dependency) between X_i and X_j; a high distance $dI(X_i, X_j)$ reflects a strong independence (low value for $R(X_i, X_j)$). On the other hand, a high estimate for Equation 3.35 describes a substantial deviation from independence (sound dependency) among X_i and X_j. Formally speaking, and analogously with $I(X_i, X_j)$, if $R(X_i, X_j) = 1$, then X_i and X_j are truly dependent; if $R(X_i, X_j) = 0$, then X_i and X_j are independent ($p(X_i, X_j) = p(X_i)p(X_j)$); if $0 < R(X_i, X_j) > 1$, then then X_i and X_j have certain degree of dependency; and lastly, if $R(X_i, X_j) > R(X_i, X_k)$, $X_i, X_j, X_k \in X$ and $i \neq j \neq k$, then X_i and X_j have a greater dependency than X_i and X_k [39].

Assume that $c \in \Re$ is the punctual result of either $dI(X_i, X_j)$ or $R(X_i, X_j)$. Table 3.4.1 describes some properties of these two dependency estimators.

Conclusively, given a collection of random variables the maximum distance for $dI(X_i, X_j)$ is the minimal dependency for $R(X_i, X_j)$. These two estimators and some other normalized versions of mutual information were pre-

Table 3.2: Relationship between the $dI(X_i, X_j)$ and $R(X_i, X_j)$ metrics.

$c \in \Re$	$dI(X_i, X_j) = c$	$R(X_i, X_j) = c$
0	Total Dependency	Total Independence
1	Total Independence	Total Dependency

sented in [57]. The interdependency redundancy measure was chosen as the dependency metric for effects of the presented work. This decision is founded in the following assumptions: $R(X_i, X_j)$ is symmetrical to $dI(X_i, X_j)$, I preferred to semantically sustain a maximization of dependency rather than a minimization of independence.

Although, the interdependency redundancy measure is not as accurate as the mutual information distance defined in Equation 3.34 it is simpler and conclusively, more robust than simply using mutual information. The last argument is valid since $R(X_i, X_j)$ is a relative true metric. The interdependency redundancy measure is an information theory based metric which is independent of feature or background information. Hence, it can be universally applied to either continuous or discrete domains.

Chapter 4

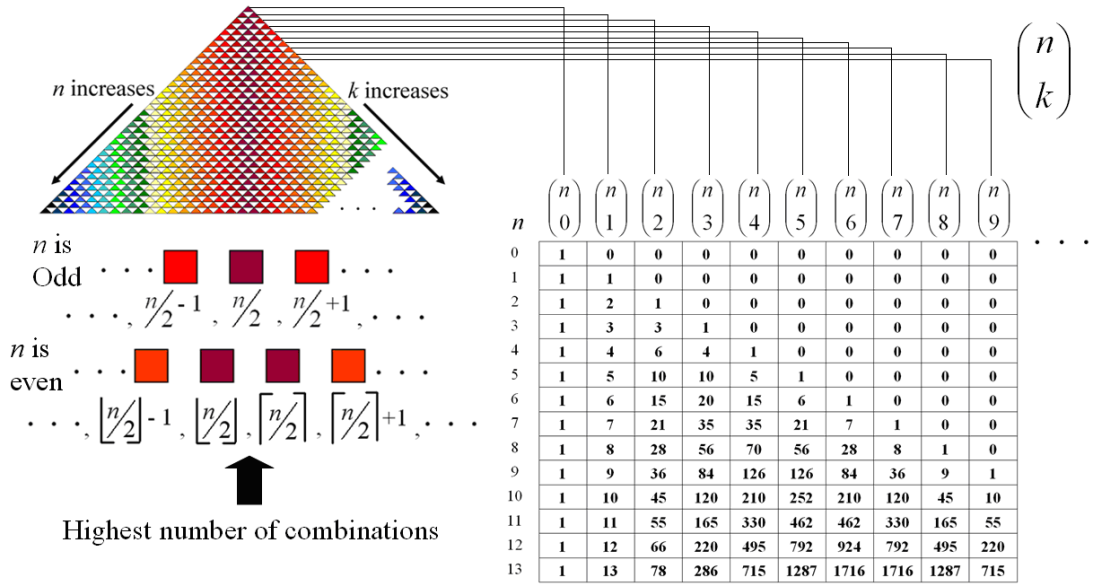
Dependency Graphs from Data

We investigated in Chapter 3 several possible metrics, distances and information theory measures that could help us to reveal dependencies among variables. We concluded that the i.r.m. is a good estimate to evaluate dependencies among a set of variables. In this Chapter, we discuss the importance of using dependence graphs for describing a set of random variables.

We are interested in gaining reduced knowledge from data. Discovering all necessary interactions between groups of variables is not a trivial task. A sound attempt is to obtain a graph which describes the domain's nature. Then, another sectioning method could partition such graph in order to generate a set of subsets of χ. On the other hand, we could also try some information theory tools such as the calculation of mutual information for more than 2 variables.

We could try to experiment with combinations of k-subsets (for some small k) of variables and then keep those groups of variables that present higher estimates. However, the computation of many combinations is a slow task. Therefore, it is not reasonable to test too many combinations. We can see in Figure 4.1, that the number of possible combinations grows polynomially (being the k-subsets between n and $k = \frac{n}{2}$ the highest possible number

of combinations).



n	$\binom{n}{0}$	$\binom{n}{1}$	$\binom{n}{2}$	$\binom{n}{3}$	$\binom{n}{4}$	$\binom{n}{5}$	$\binom{n}{6}$	$\binom{n}{7}$	$\binom{n}{8}$	$\binom{n}{9}$
0	1	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
2	1	2	1	0	0	0	0	0	0	0
3	1	3	3	1	0	0	0	0	0	0
4	1	4	6	4	1	0	0	0	0	0
5	1	5	10	10	5	1	0	0	0	0
6	1	6	15	20	15	6	1	0	0	0
7	1	7	21	35	35	21	7	1	0	0
8	1	8	28	56	70	56	28	8	1	0
9	1	9	36	84	126	126	84	36	9	1
10	1	10	45	120	210	252	210	120	45	10
11	1	11	55	165	330	462	462	330	165	55
12	1	12	66	220	495	792	924	792	495	220
13	1	13	78	286	715	1287	1716	1716	1287	715

Figure 4.1: Possible number of combinations between a set of n elements and a given integer number k.

Before we proceed with our investigation, it is important to notice that through this work, we will tackle discrete random variables. We will denote the set of all variables with the symbol χ that is the set of the n random variables $\chi = \{X_1, X_1, \ldots, X_n\}$ from a given domain. Following, several dependency graphs are explained in detail. A final note is that, for any graph that is presented through this work, every node corresponds to a random variable. Every edge corresponds to a "sound" interaction or dependency between pairs of variables. The weight that is attached to an edge in a dependency graph is the degree of dependency or interaction between those two variables (we chose the i.r.m. measure to denote weights in any dependency graph because of the reasons explained in the final section of Chapter 3).

4.0.2 The N-Cut Dependency Graph

Another idea is to compute the complete graph K_χ over the set χ (i.e. by using mutual information or the i.r.m.), sort its weights decreasingly; and then, add the N edges whose weights are maximal. This method is highly intuitive but also highly naive; in practice, many domains will present a small group of variables that are highly correlated, but many others will be isolated since they are not statistically significant. Figures 4.2, 4.3 and 4.4 present an example of this method when applied to dataset generated from the Alarm BN[1], in every case we set N to 3 different values; we selected the $|\chi|$, $2|\chi|$ and $3|\chi|$ edges with highest weights.

The N-cut approach is an agglomerative method, which adds arcs given a parameter N. However, this method lacks of the universality property because depending of the domain in study we may require to add more or less arcs. Figure4.5 shows the weight distribution over the complete graph built for the Alarm domain (notice how it is not intuitive to select a threshold N).

The granularity parameter N has to be regulated according to the domain. For example, in Figure 4.2 we can see that if we set N to a very small value then we end up with few components and many isolated variables (from the set of 37 variables only 23 variables were assigned to 5 groups).

Secondly, if we slightly increment N to 74 (Figure 4.3) we can observe that the number of considered variables increased to 29 (in 4 groups). Nevertheless, 8 variables are still not considered. Finally, in Figure 4.4 we aimed to attach a considerably higher amount of edges (111). In this case we ob-

[1]The ALARM BN was firstly introduced by Beinlich et al. [58] for monitoring patients in intensive care. This Bayesian network is preferred for testing deterministic algorithms due to its simplicity. In this Chapter, we will show examples of dependency graphs that were built by using generated data from this BN with a sample size $\Omega = 10,000$.

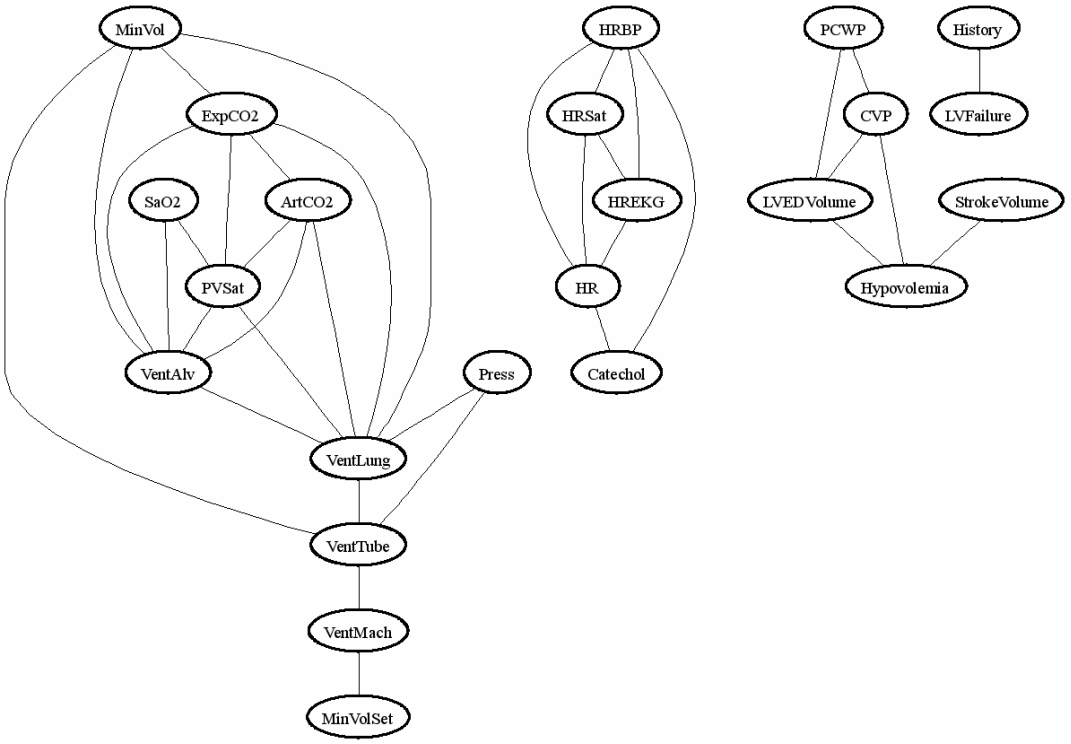


Figure 4.2: The N-cut method. The parameter N was set to $|\chi| = 37$.

tained a single group with 31 variables. Obviously, from this scenario we can foresee that if we continue incrementing N, we may cover all 37 variables soon. However, having such big groups is impractical, since they do not reveal simplified settings from the domain.

Judging by the previous facts, we can have the certainty that finding a domain's topology is not a straightforward process. It is convenient to investigate some other paradigms that could help us to encapsulate the most relevant dependencies among the full collection of variables. In the next section we present a key feature in our work, a dependency graph called the Maximum Spanning tree.

4.0.3 The Maximum Spanning Tree

The Maximum Spanning (MAST) tree was firstly introduced by Chow and Liu in [59] and has been through fully investigated in [60]. This structure was one of the first attempts to represent probabilistic distributions by using a graphical model. In concrete, the Maximum Spanning tree has the objective of approximating a joint probability distribution $P(X_1, X_2, \ldots, X_n)$ among n variables. We shall note that a MAST is the smallest graph that optimally approximates the probability distribution between the variables. In other words, a Maximum Spanning Tree (MAST) $MAST = (VMAST, EMAST)$ is a set of nodes $VMAST = \{X_1, \cdots, X_n\}$ connected by a set of edges $EMAST = \{e_{1,j}, \cdots, e_{y,n-1}\}$. Each edge $e_{i,j}$ is attached with the weight $R(X_i, X_j)$. The MAST provides a raw dependency graph. Fig. 4.6 describes a MAST construction procedure.

In this procedure, the i.r.m. $R(X_i, X_j)$ is computed for all pairs of variables (for $l = 1$ to Ω samples) and an adjacency matrix CA is built (line 1). The complexity of this task is in the order of $O(n^2)$. In lines 3-6, a modified version of the Kruskal's algorithm is used to build the tree (the original Kruskal's algorithm for finding the minimum Spanning tree sorts the weights decreasingly instead of increasingly in line 4). We used an union-finder data structure [61] and a sorted list for adding and removing arcs. The complexity of this procedure is in the order of $O(n \log n)$.

The MAST has been used for classification of images, for several tasks in pattern recognition, and it is still considered a good probabilistic model for encoding loose constraints. An example of the MAST is shown in Figure 4.7 for the Alarm Bayesian network[58]. We used 10,000 samples and the i.r.m. to generate it. Notice how the adjacent variables in the MAST turn out to be also adjacent variables in the BN (Appendix A).

The MAST is the smallest connected graph that "optimally" approximates the joint probability of the domain[2]. On the other hand, if we use the MAST not as a graphical model, but as a dependency graph that reveals a sound amount of dependencies; then it could not be enough to represent all the valuable dependencies.

The constraint of leaving only $n-1$ edges restrict also those other weights that may provide helpful dependencies. Thus, many important relationships are lost because of this restriction. It might be convenient to augment this tree in order to gain information about the most important dependencies. Nevertheless, its value relies in the fact that it can be easily partitioned for domain reduction. In the next Chapter we introduce several partitioning methods (some take the MAST as the dependency graph to be partitioned), and in the next sections we propose several extensions to the MAST.

4.0.4 Beyond the MAST

The MAST by itself is a good candidate to depict the minimal connectivity among all variables in χ. Nevertheless, we show in this section several augmentation procedures in order to recover more dependencies: Minimal expansion, maximal expansion, average expansion and the Maximum Spanning Forrest. These add-ins could help to direct further sectioning algorithms in order to discover smaller groups of variables.

Minimal Expansion

This method takes the MAST and the complete graph as inputs and then for every variable X, the method finds its arc with lowest weight w, then we find

[2]Although BNs are known to be the best graphical models for graphical probabilistic modeling.

the edge e' (from the complete graph) with the highest weight w' connecting another variable X' to X (X' is not initially adjacent to X in the MAST). Finally we add e' to the dependency graph iff w' ¿ w. This method adds very few edges, (around 30 percent of the $n-1$ initial arcs). Figure 4.8 shows an example of such expansion.

Maximal Expansion

The maximal expansion method takes the MAST and the complete graph as inputs and then, for every variable X, the arc with lowest weight w is detected; then we find the set of edges E' with weights W' connecting the set of variables $X'_1, X'_2, \ldots, X'_l$ to X (the variables $X'_1, X'_2, \ldots, X'_l$ are not initially adjacent to X in the MAST), finally we add E' to the dependency graph iff $w < w' \in W'$. This method adds the highest number of arcs producing a dependency graph with around 4(n-1) edges (Figure 4.9).

Average Expansion

The average expansion receives the same inputs as the previous methods and again, for every variable X, the method finds the average weight w of the weights attached to the arcs connecting X. Following, we find the set of arcs E' with weights W' connecting the set of variables $X'_1, X'_2, \ldots, X'_l$ to X (the variables $X'_1, X'_2, \ldots, X'_l$ are not initially adjacent to X in the MAST). At the end, we add E' to the dependency graph iff $w < w' \in W'$. This method adds around $\frac{(n-1)}{2}$ extra arcs to the MAST ($(n-1) + \frac{(n-1)}{2}$ edges in total). Figure 4.10 shows an example of the average expansion. 4.9).

Maximum Spanning Forrest

The Maximum Spanning Forrest (MSF) provides a more robust picture of the domain's nature. Thus, it reinforces the dependencies between highly correlated variables by using small expansions. Therefore, it fulfills the two constraints of minimal connectivity with maximal weights. This method is basically the MAST approach but applied for N iterations. After one MAST is computed, then its weights and arcs are removed from the complete graph (from which the spanning forest is built). At the end, the $MSF - N$ is the smallest dependency graph with N inner trees that maximize the total weight of the graph. It was seen empirically, that N should be kept to a small value, and N may vary according to the domain in study. Figures 4.11 and 4.12 show the second and third tress that were obtained after the initial MAST shown in Figure 4.7 was calculated.

All the previous dependency graphs offer a sound interpretation of the domain's topology. Moreover, the decision of considering a given dependency graph will depend on the complexity of the domain in study. Conclusively, an specialized clustering method has to be applied for each dependency graph (since clustering or sectioning methods that aim partitioning trees are logically not the best options for partitioning highly connected graphs). In this work we chose the complete graph and the MAST for further grouping of variables. The reason for the later is that the complete graph provide full information regarding pairwise interactions. On the other hand, the MAST is a dependency graph that can be rigorously partitioned without loosing generality. The next Chapter explains in detail some relevant clustering methods used in this investigation.

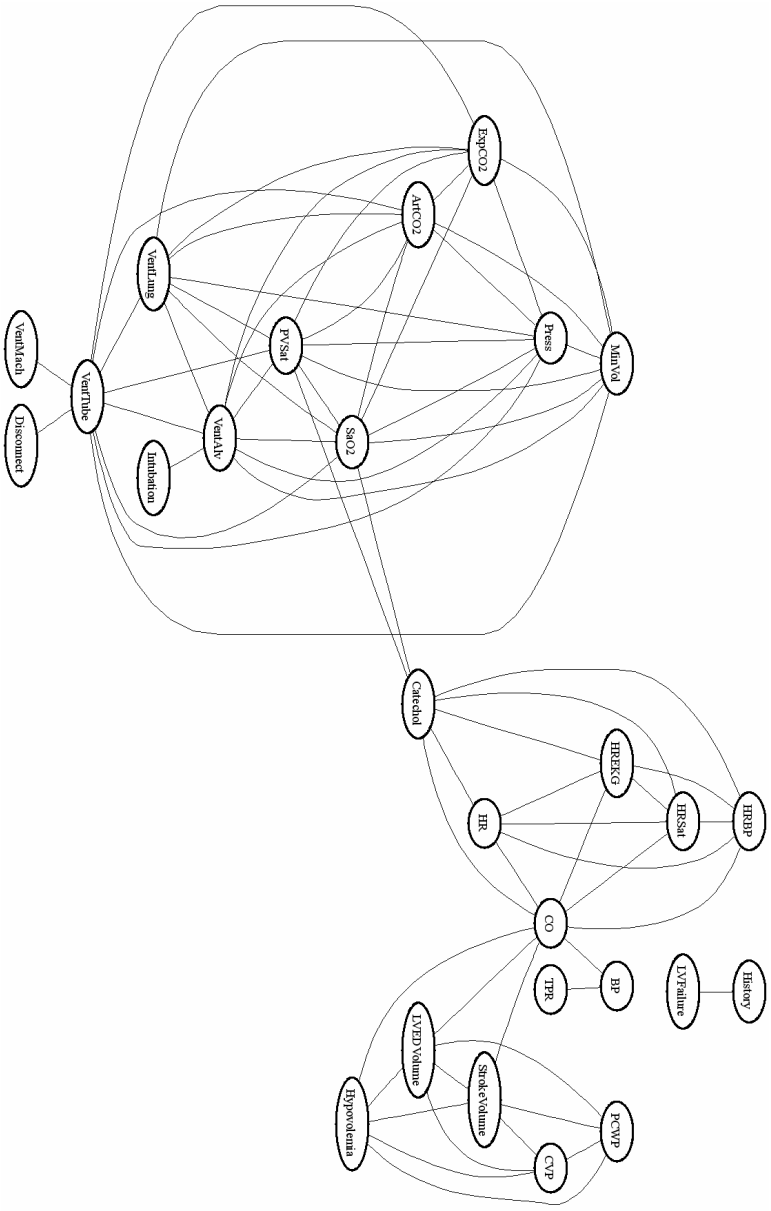


Figure 4.3: The N-cut method. N was set to $2|\chi| = 74$.

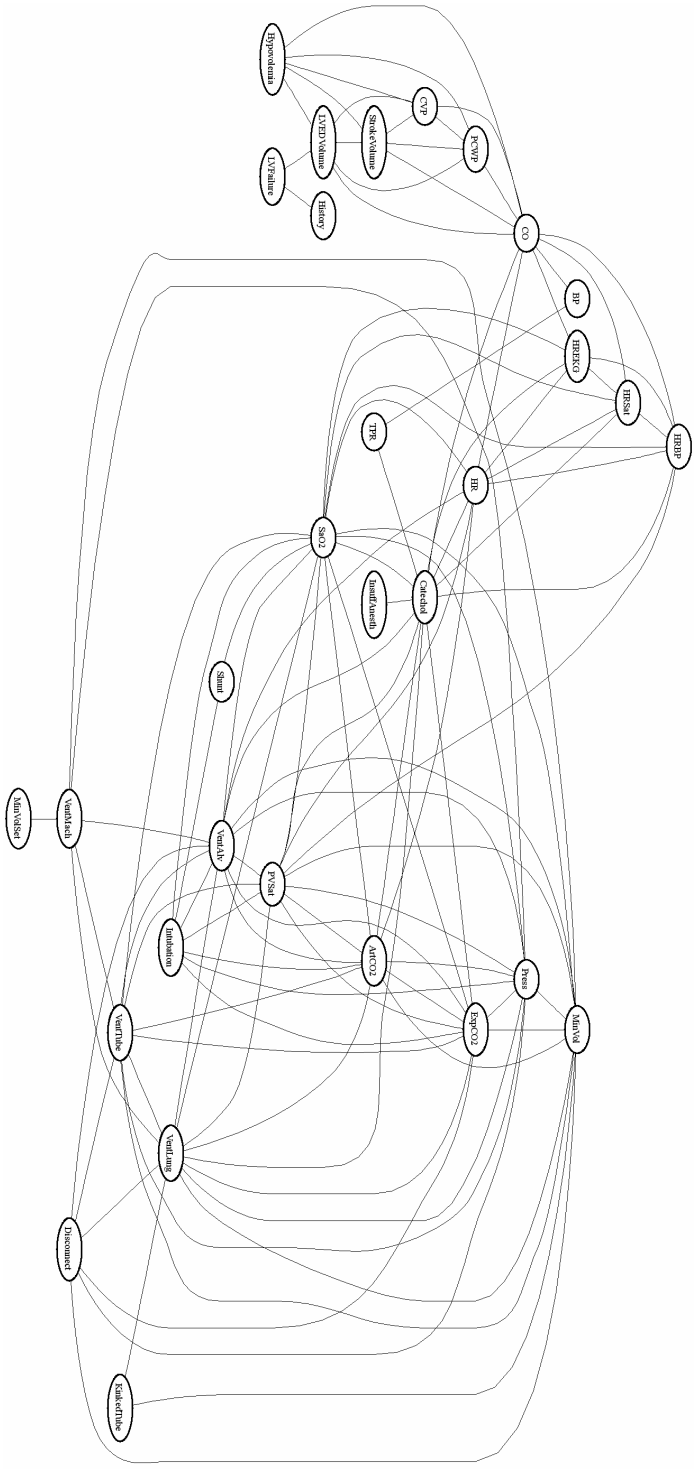


Figure 4.4: The N-cut method. N was set to $3|\chi| = 111$.

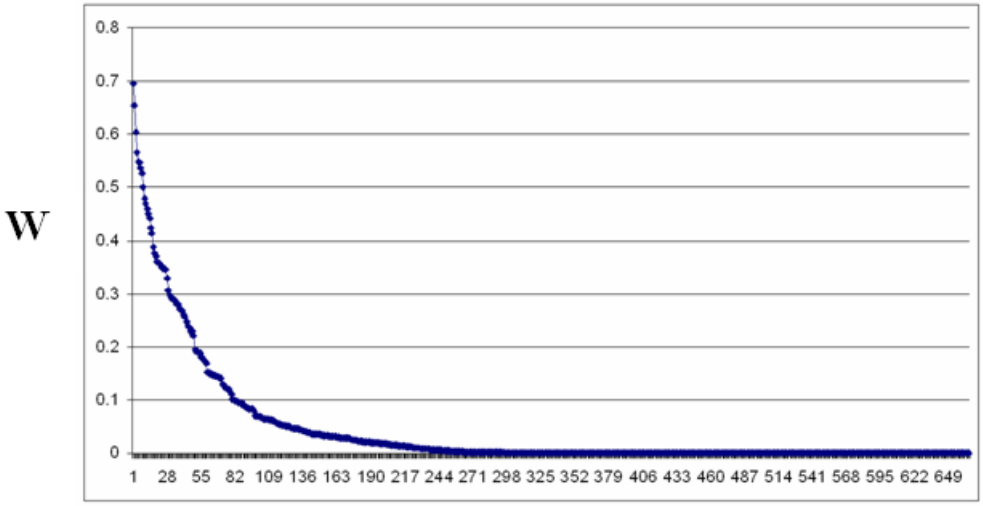


Figure 4.5: The weight distribution in the complete graph across all edges for the Alarm domain.

MAST Construction Algorithm

Input: Data $D = \{a_{1,l}, \cdots, a_{n,l}\}$

Output: $MAST = (VMAST, EMAST)$, MA

1: Compute a complete Graph $G = \{\chi, E\}$ with weights $CA = \{ca_{i,j} = R(X_i, X_j) | i, j = 1, \cdots, n \text{ and } i \neq j\}$

2: $k = 0$, $VMAST \leftarrow \chi$, $EMAST \leftarrow \emptyset$, $MA \leftarrow \emptyset$

3: Sort E decreasingly according to the weights in CA

4: **FOR** $e_i \in E$ **AND** $k < |\chi|$ **DO**

5: **IF**$(e_i \in EMAST)$ do not create a cycle in $MAST$ THEN

6: $EMAST \leftarrow (e_i \cup EMAST)$, $MA \leftarrow (ca_{i,j} \cup MA)$, $k = k + 1$

Figure 4.6: Identifying a maximum spanning tree from Data D.

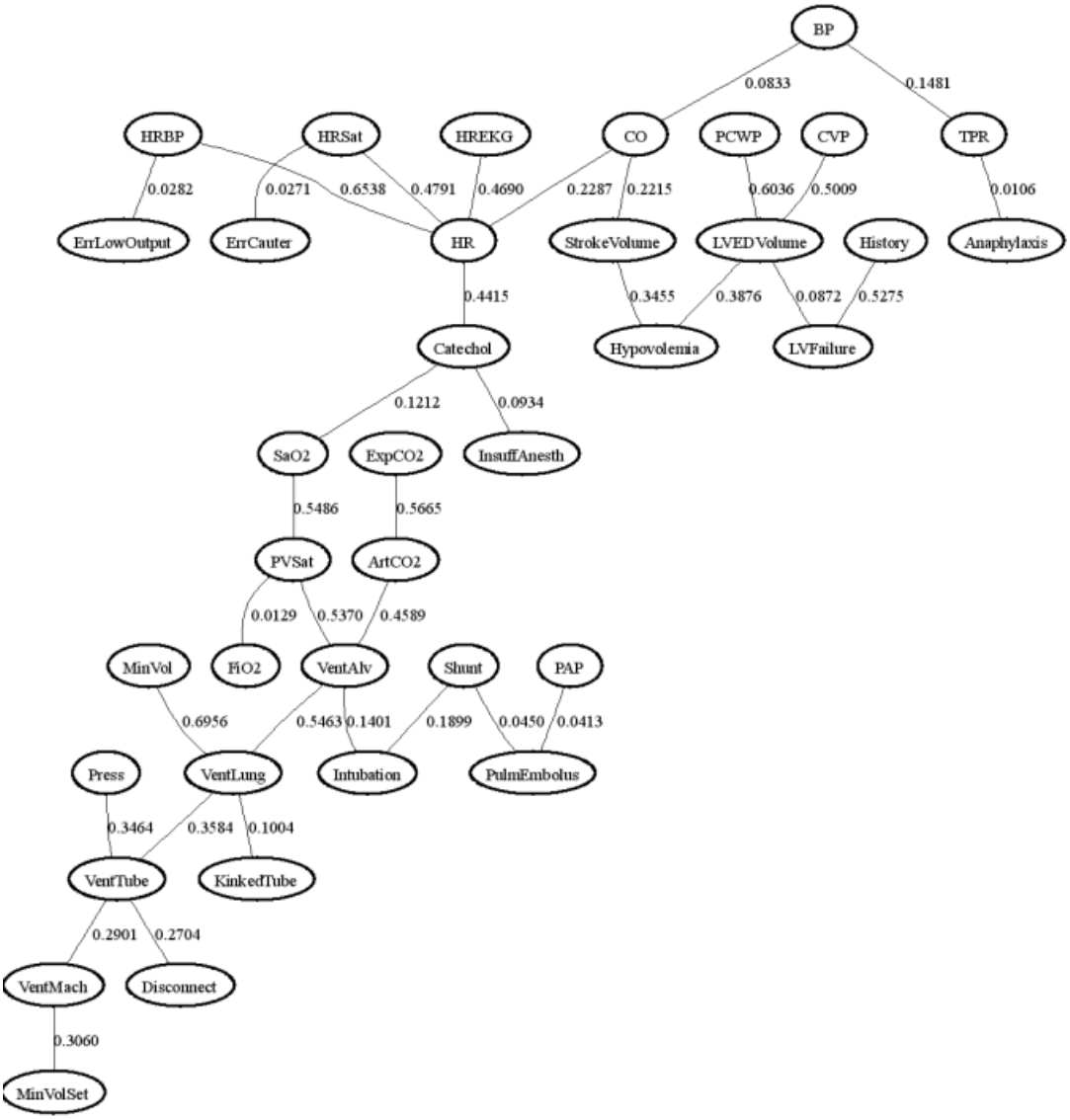


Figure 4.7: The MAST for the Alarm BN.

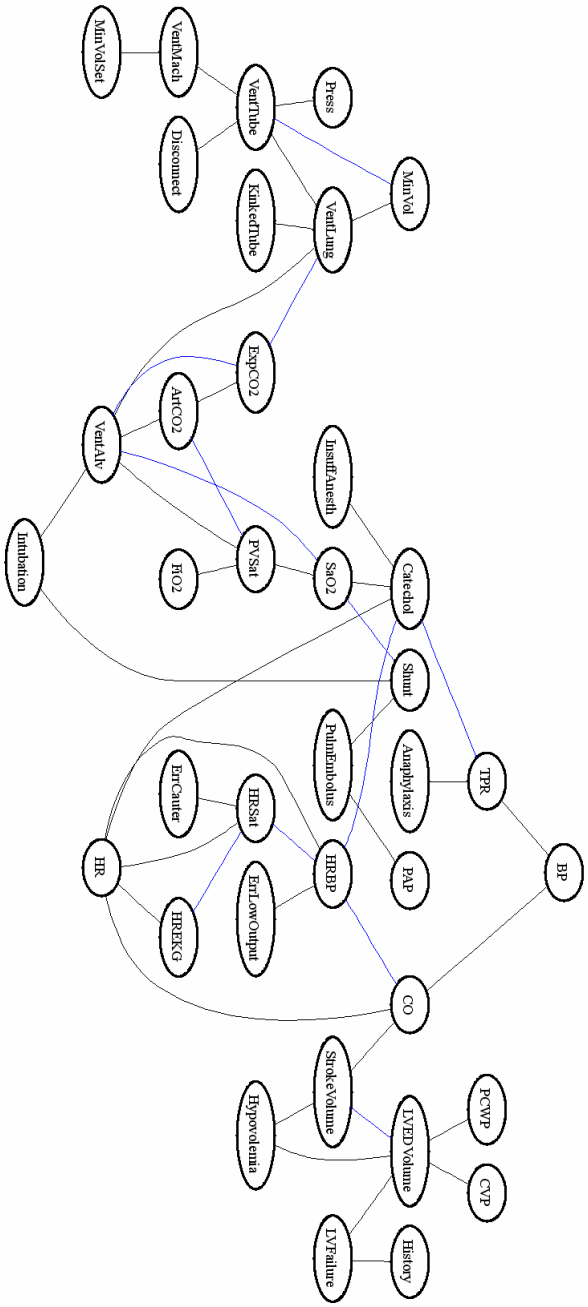


Figure 4.8: The minimal expansion for the MAST in the Alarm BN.

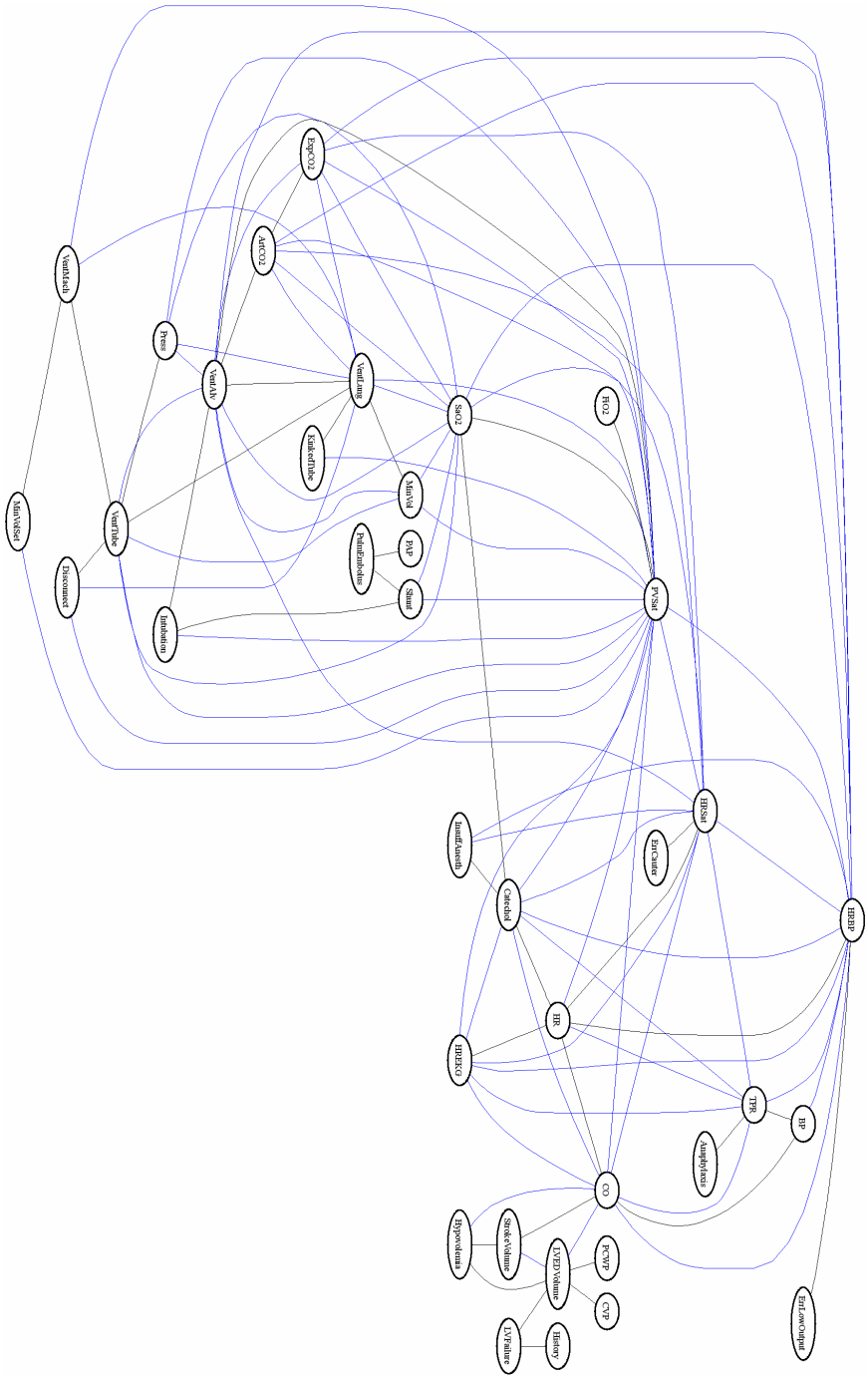


Figure 4.9: The maximal expansion for the MAST in the Alarm BN.

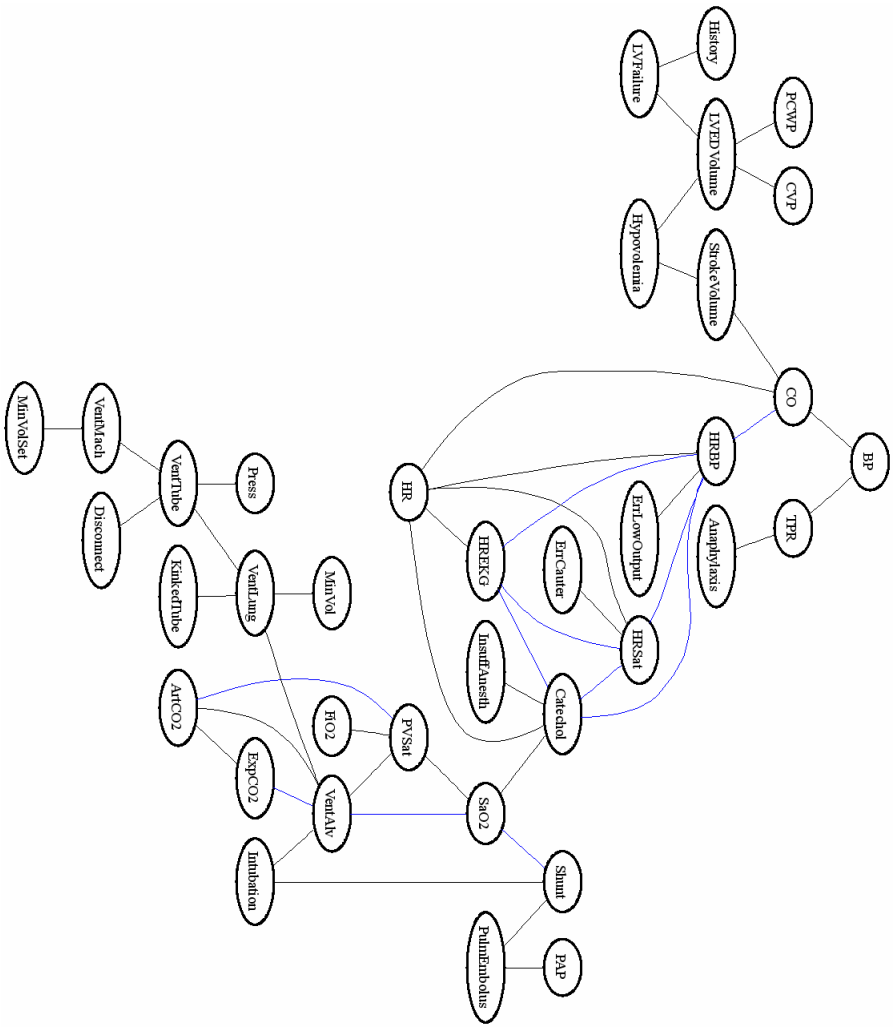


Figure 4.10: The average expansion for the MAST in the Alarm BN.

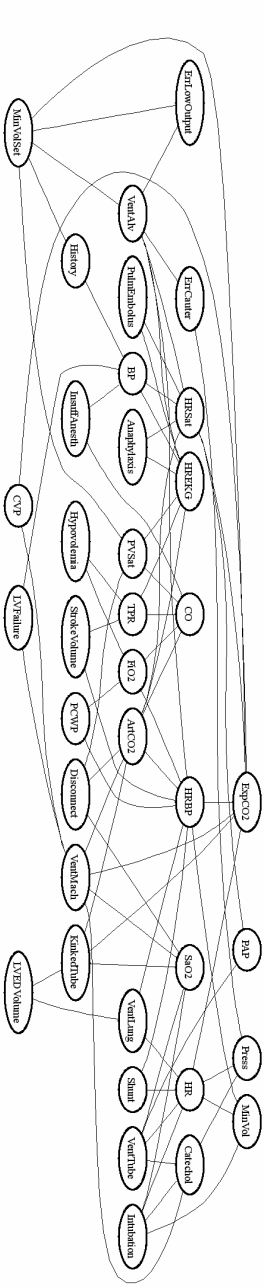


Figure 4.11: The second MAST in the Alarm BN.

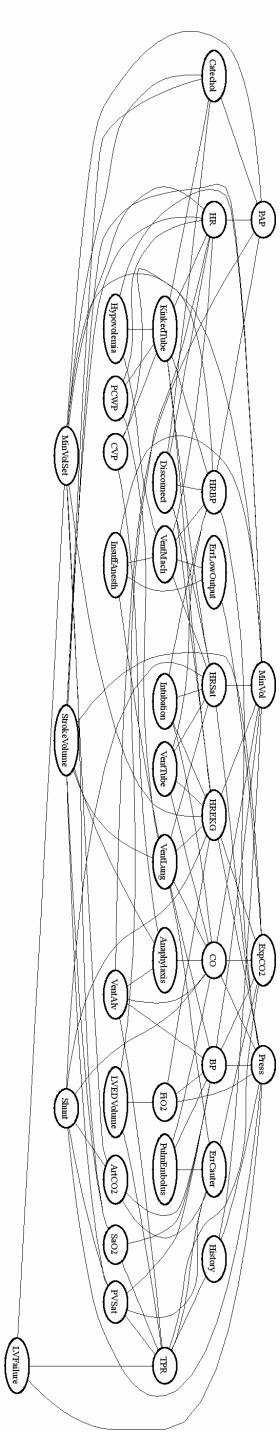


Figure 4.12: The third MAST in the Alarm BN.

Chapter 5

Discovering Local Components

How can we define a group of highly correlated genes while examining gene expression data? How can we find those relevant features in a dataset having numerous variables? How could we define a set of classes over a collection of observations? Basically, all these questions have been the fundamental motivation for one of the most popular branches of data mining; data clustering.

Data Clustering is probably one of the best known techniques of data mining. This task is one of the most important unsupervised learning problems that computer scientists, statisticians and mathematicians have tried to develop and improve for years. For more than two decades, clustering has taken special interest between the scientific community and it is probably in the jargon of other professionals that have no direct relationship with machine learning or even computer science. The reason for the later relies beneath its clear definition and interpretation. In general, we can define clustering as the task of detecting groups of points, entities or elements that hold a substantial similarity.

There exist two potential fields in which clustering has been extensively studied in the past. Firstly, data clustering is a data mining operation that help us to define relationships between samples over a given dataset. These

groups of similar objects are then labeled as clusters; then they can be used either for data analysis directly or to define classes which describe the domain upon several characteristics. Probably the second widest use of clustering in the past years has been the task of selecting genes (variable selection) in Bioinformatics. Both approaches rely on a golden principle, the optimization of an objective function.

In the case of data clustering (and according to [62]), this task can be classified in for major paradigms:

- Model based methods.
- Hierarchical methods.
- Partitioning methods.
- Density estimation methods.

Besides of the previous classification, clustering approaches can also be independently classified in other terms. They can be either exhaustive or not exhaustive methods (exhaustive procedures aim to add every possible object to a given cluster, whereas in the second case some variables could be left aside isolated). Moreover, a given clustering algorithm can produce disjoint or overlapping clusters (By using the same notation and lexems than in set theory we can define that a disjoint set of clusters $\{C_i, C_j\}$ have no element in common $\exists_{c \in C_i} c \subseteq (C_i \wedge C_j)$, in the sacond case they could share one or more variables such that $C_i \wedge C_j \notin empty$).

Every clustering algorithm in literature has been developed with the aim of solving very specific problems. The construction of clustering algorithm is normally ad hoc; logically, two clustering algorithms that present different aims become incomparable in the same terms.

For the effects of this project we aim for the second class of clustering, namely variable selection and more specifically, attribute clustering. We also reduced our search for clustering algorithm to the class of partitioning techniques. The later was done because of the focus of this work, that is the discovering of highly correlated random variables for simplified analysis of a given domain. We also adopt the assumption that all variables can be seen as points in an euclidean space because we have complete information regarding pairwise proximities. Thus, the use of a true metric is remarkably helpful since the partitioning becomes intuitive.

Before we continue with our discussion it is convenient to introduce the most well known partitioning clustering algorithm, the k-means [63]. This algorithm is intuitively based on the discovery of k different clusters among a collection of n points $p_1, p_2, \ldots, p_n$ by grouping each element to its nearest mean $m_1, m_2, ..., m_k$ (Initially the means are either selected or chosen at random) such that their distance $d(p_j, m_i)$ is minimal. Once that each point has been covered in one cluster a new "center of mass" or mean is again recalculated in order to define a geometrical center between that local collection of points. Finally, the two previous instructions are repeated until convergence. Algorithm 5.1 present an abstraction of the k-means algorithm.

The k-means algorithm can actually be formulated in more specific terms. We say that k-means algorithm aims to find a superset of points C over every element of P by maximizing the following objective function:

$$W = \underset{p_j \in P, m_i \in M}{\operatorname{argmax}} \left(\sum_{j=1}^{n} \sum_{i=1}^{k} d(p_j, m_i)\right) \tag{5.1}$$

A further extension of the k-means algorithm is the k-modes or k-medoids algorithm that is basically similar in nature to algorithm 5.1 except that in

K-means Algorithm

Input: A collection of points $P = \{p_1, p_2, \ldots, p_n\}$ and means $M = \{m_1, m_2, ..., m_k\}$
Output: A set of clusters $C = \{C_1, C_2, \ldots, C_k\}$

1: WHILE $M^{old} \neq M$
2: $M^{old} = M$
3: Assign each element p_j to C_i iff $d(p_j, m_i)$ is minimal.
4: Once that all elements in P have been covered recalculate the means M

Figure 5.1: A general description of the k-means algorithm.

this case instead of using virtual means we define an elements in P as the modes or centers for each cluster. This, algorithm present a very intuitive and solid foundation for variable selection since the center of each cluster has special relevance (it is the variable with the highest similarity with any other variable in its group). In the next section we will study in detail an instantiation of the k-modes algorithm and we will explain its relevance to our study.

Up to this point we have introduced the necessary background to guide our focus into one of the objectives of this study, the selection of clusters of highly dependent random variables. Clearly, we can reduce the problem of discovering local components of similar random variables into a clustering problem, and more specifically into a graph partitioning problem. For the next sections on this chapter we will assume that the data has been statistically measured in terms of the i.r.m and that each pair of variables X_i and X_j $(i \neq j)$ hold a similarity distance $d(X_i, X_j)$ that is equal to $R(X_i, X_j)$. Therefore, the complete set of distances between all the possible combinations of variables χ can be seen as weights W attached to edges E in a complete undirected graph $K_n = (\chi, E)$.

5.1 Attribute Clustering Methods

Two paradigms of graph partitioning were taken in order to find clusters of random variables. The first technique is the attribute clustering algorithm based in a k-modes algorithm and it is presented in the next section. Later in this chapter we will also examine another sound method for performing clustering of attributes based in the partitioning of Minimum and Maximum Spanning Trees.

5.1.1 Attribute Clustering Algorithm

The novel attribute clustering algorithm (ACA) has its root in the class of the k-medoid [64] approach. The k-modes algorithm [39] is basically an implementation of the k-medoid algorithm but replacing euclidean distance by the i.r.m measure. Given the set of variables $\chi = \{X_1, X_2, \ldots, X_n\}$ and an integer k, the objective of ACA is to find a disjoint set of clusters $C = \{C_i | (i = 1, \ldots, k) \wedge (\forall_{i \neq j} C_i \cap C_j = \emptyset)\}$ with centers $O = \{o_i | i = 1, \ldots, k\}$ which has the maximal global weight W^C which is presented in Equation 5.2:

$$W^C = \sum_{i=1}^{k} \sum_{j=1}^{|C_i - o_i|} w_{j,o_i} \tag{5.2}$$

Ultimately, the aim in this algorithm is to iteratively find the right centers O. Figure 5.2 presents the attribute clustering algorithm.

ACA receives a complete graph K_n and two positive integers: The number of Stars k and the maximal number of iterations q.

The algorithm works in the following order: Firstly, in the initialization phase (line 1) a group of centers is selected and stored in the auxiliary set O^{new}; the set $Rand_k^{\chi}$ is an artificially generated set of k variables from χ

Attribute Clustering Algorithm (ACA)

Input: $K_n = (\chi, E)$, $k, q \in \mathbf{Z}^+$
Output: $C = \{C_1, C_2, \ldots, C_k\}$

1: $O^{new} = Rand_k^{\chi}$, $O = \emptyset$
2: **WHILE** $O^{new} \neq O$ **AND** $r \leq q$
3: $O = O^{new}$
4: $C = \emptyset$, $C_1 = \emptyset, C_2 = \emptyset, \ldots, C_k = \emptyset$
5: $O^{new} = \emptyset$, $o_1^{new} = \emptyset, o_2^{new} = \emptyset, \ldots, o_k^{new} = \emptyset$
6: **FOR** $i = 1$ to k
7: $C_i \Leftarrow o_i$
8: **FOR** $i = 1$ to $|\chi - O|$
9: $C_j \Leftarrow X_i$ iff $w_i = argmax_{o_j \in C_j}(w_{X_i,o_j})$
10: **FOR** $i = 1$ to k
11: $C \Leftarrow C_i$
12: **FOR** $i = 1$ to k
13: $o_i^{new} = X_j$ iff $w_j = \underset{X_j \in C_i}{\text{argmax}}(\sum_{h=1}^{|C_i - X_j|} w_{X_j,X_h})$
14: $O^{new} \Leftarrow o_i^{new}$
15: $r = r + 1$

Figure 5.2: The Attribute Clustering Algorithm.

taken at random. Immediately, the set of working centers is obtained (line 3) and all clusters are renewed (line 4), the auxiliary O^{new} is set to empty for further center assignations and all clusters are generated (lines 6-7) by adding their centers. In the assignment phase (lines 8-9), a variable X_i is assigned to a cluster C_j iff the weight $w_{X_i,o_j} > w_{X_i,o_h}$ is the highest for all $o_j, o_h \in O$ such that $o_j \in S_j \wedge o_h \in C_h$. Following, the clusters are assigned to C (lines 10-11). In the next phase (lines 12-14), a new center o_i^{new} in all clusters $C_i \in C$ is calculated; such that $o_i^{new} = X_j$ for all $X_j, X_h \in C_i$ iff $\sum_{l=1}^{|C_i - X_j|} w_{X_j,X_l} > \sum_{l=1}^{|C_i - X_h|} w_{X_h,X_l}$.

The complexity of the attribute clustering algorithm can be easily esti-

mated by considering a total of n variables in χ, k clusters, q iterations and a total of p samples ($p = \Omega$).

First of all, there is the need to receive a complete graph with all pairwise arcs; the computation of such a structure is in the complexity of $O(n^2p)$. Following, we consider only the most phases which dominate the complexity of the algorithm. First, in the assignment phase (lines 8-9) it takes $O(k(n-k)) \in O(kn)$ operations to assign all free variables to the clusters. Then, when the new centers are calculated for all clusters (lines 12-14), it takes $O(t^2)$ steps to decide the highest weight W^{C_i} of one cluster C_i having that t is the highest cardinality of the Stars. Thus, it takes $O(kt^2)$ operations to discover all centers $o_i \in O$.

In conclusion, it takes a time in the order $O(k(n+t^2)q) \in O(knt^2q)$ to output the set of clusters C if we do not compute the complete graph K_n. If K_n is not provided then ACA has a time complexity of $O((n^2p)+(knt^2q)) \in O(kn^2pq)$. ACA is a smooth process, simple but robust enough at the same time; it has shown convincing results for gene selection and micro array analysis in two datasets describing the gene expression levels of a form of leukemia [35] and colon cancer [34].

5.1.2 Standard Euclidean Minimum Spanning Tree

The k-modes approach is a straightforward approach to detect the clusters based in some center variables. However, the k-modes algorithm attempts to partition a complete graph K_n containing n random variables. Thus, it results convenient to choose a looser dependency graph to be partitioned. One of the most popular graphs for clustering is the Minimal or Maximal Spanning Tree (MST and MAST) presented in Chapter 4.

Clustering algorithms based in partitioning MST (MAST) are commonly

applied to gene expression analysis and image processing. One application of this method was successfully devised by Xu et. al in [65, 66] in bioinformatics. An interesting application of maximum and minimum spanning trees to image color clustering can be found in [67].

A minimalistic topology of the domain is relevant since we are interested in finding groups of non distant variables seen as points in an euclidean space. Thus, as previously stated, the MST creates a backbone in which only the highest correlations are recovered. In the MST one arc depicts only the most dependent variables in the domain, thus the partitioning becomes intuitive, and now the question is to decide which edges are appropriate to remove from this tree.

The MST-based clustering algorithms are commonly based in heuristics that aim at removing a set of "inconsistent" edges from the tree. An important factor in this technique is the selection of a heuristic or process which decides what arcs are relevant (and will remain) and which edges are inconsistent with the topology and removed. These algorithms do not require a high amount of parameters to perform bisections over a tree; however, they are probably the best clustering options for domains containing points that are not geometrically well situated. Nevertheless, this class of algorithms are faster than the k-means type of algorithms at the price of quality of the solution. It is interesting to develop a fast partitioning algorithm that could provide a satisfactory result. Figure 5.3 presents a framework for MST-based clustering algorithms.

Since all algorithms that construct Minimum and Maximum Spanning Trees require an initial complete graph K_n; we will consider that all of the MST-based algorithms receive the Maximum Spanning Tree $MAST$ beforehand (we already know that in order to produce the complete graph we need

MST-based Algorithms

Input: A complete graph $K_n = (\chi, E)\}$ with weights W^{K_n}
Output: A set of clusters $C = \{C_1, C_2, \ldots, C_k\}$

1: Find a Maximum Spanning Tree MST from K_n.
2: Remove the "inconsistent" edges from MST.
3: Form the set of clusters from the remaining subgraphs from MST.

Figure 5.3: A Framework for algorithms which partition a MST.

to compute the complete set of weigths in $O(n^2p)$ where p is the number of observations $p = \Omega$). Having introduced the framework of tree based partitioning we will proceed to review several clustering algorithms based on this principle.

The first and simplest MST-based approach is called the Standard EMS clustering algorithm. The short term EMST stands for Euclidean minimum spanning tree algorithm. That is the use of a MST to divide sets of dense regions of points. The SEMST algorithm was introduced by Asano et. al in [68]. This algorithm applies the principle of separability which states that two sets of points over connected trough a MST are separated by stabbing line. In other words k sets of points can be isolated in a MST if we remove the $k-1$ "inconsistent" edges whose weight is maximal (or minimal, depending if we are employing a minimum or maximum spanning tree). The SEMST algorithm is presented in Figure 5.4.

The objective function of the SEMST algorithm is not that simple to determine since the algorithm is mostly discriminative only with respect of $k-1$ arcs. We have that, the objective is to maximize the sum of weights W^G for every subgraph $G_i = (V_i, E_i)$ (with a set of variables V_i and a set of

arcs E_i) after we delete $k-1$ arcs. Thus, the objective function becomes:

$$W^G = \sum_{i=1}^{k} \sum_{X_h, X_j \in V_i} w_{h,j} \tag{5.3}$$

Notice that for this thesis we aim to maximize the dependency between random variables; therefore, by following the previous premises we will partition the MAST in order to find the desired clusters.

Standard Euclidean Minimum Spanning Tree Algorithm (SEMSTA)

Input: $MAST = (\chi, EMAST)$, W^{MAST}, $k \in \mathbf{Z}^+$
Output: A set of clusters $C = \{C_1, C_2, \ldots, C_k\}$

1: FOR $i = 1$ to $(k-1)$
2: $I = (I \cup (X_r, X_s))$ iff $(X_r, X_s) \notin I \wedge (X_r, X_s) = \underset{w \in W^{MAST}}{\operatorname{argmin}}(w)$
3: FOR $i = 1$ to I
4: $e_i = (X_r, X_s)$
5: $EMAST = (EMAST - e_i)$
6: **IF** $i = 1$
7: $G_1 \Leftarrow (X_r \cup VR_{X_r}^{MAST})$
8: $G_2 \Leftarrow (X_s \cup VR_{X_s}^{MAST})$
9: $G \Leftarrow G_1$, $G \Leftarrow C_2$
10: **ElSE**
11: **FOR** $h = 1$ to $|G|$
12: **IF** $e_h \in EMAST_h$
13: $G_1 \Leftarrow R_{X_r}^{MAST}$
14: $G_2 \Leftarrow (X_s \cup VR_{X_s}^{MAST})$
15: $G = (G - G_h)$
16: $G = (G \cup G_1)$, $G = (G \cup G_2)$
17: FOR $i = 1$ to $|G|$
18: $C_i = V_i$ having that $G_i = (V_i, E_i)$

Figure 5.4: The Standard Euclidean Minimum Spanning Tree Algorithm

The SEMST algorithm receives the Maximum Spanning Tree $MAST$ and a number k and it initially stores in I the "inconsistent" edges having

minimal weights (lines 1-2) attached to edges $EMAST$. Then, the SEMST iteratively bisects the $MAST$ (lines 3-16) selecting each arc $e_i = (X_r, X_s)$ in I, and in the first iteration it separates the tree into two disjoint clusters $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ by pulling the reachable subgraphs R_{X_r} and R_{X_s} ($R_{X_r} = (VRX_r, ERX_r)$ where VRX_r and VEX_r are the set of nodes and edges that connect X_r,note that $X_r \in VRX_r$). Once that we have the two initial subgraphs G_1, G_2 the algorithm continue pruning arcs but now among the existing subgraphs (lines 11-16). At the end we say that each set of variables V_i from $G_i = (V_i, E_i)$ in every subgraph in $G_i \in G$ correspond to one cluster C_i (lines 17-18).

The complexity of the SEMST algorithm is trivial compared to other clustering algorithms, since it takes $O(n \log n)$ to construct the Maximum Spanning Tree. If we use the kruskal algorithm to build the initial tree $MAST$ then we already have sorted arcs according to their weights W^{MAST}, in such case it will take constant time $O(k-1)$ to remove the "inconsistent" edges (lines 1-2). For the assemble of clusters (lines 3-16) it may take at most a time of $O(kt)$ whereas t is the highest number of variables in a subgraph $G_i \in G$.

Clearly, the SEMST algorithm seems to be a simplistic but not robust algorithm for attribute clustering. Theoretically, we have that the instantaneous deletion of $k-1$ edges might produce clusters that are too small in order to acquire any knowledge whatsoever. Indeed, if we have that every weight is different, then some variables that are the least correlated in the domain will suffer the risk of being discriminated and isolated in clusters by themselves.

5.1.3 The Maximum Cost Spanning Tree Algorithm

In order to avoid the problems related with the SEMST approach we developed the Maximum Cost Spanning Tree Algorithm (MCST). The algorithm works exactly in the same fashion as SEMST. However, we bias the search of those "inconsistent" edges by substituting the weights attached to the arcs for costs. Bang Ye et. al describe in [] a class of euclidean trees, namely the Minimum Routing Spanning Trees [69]. These trees are widely used in network design. In these case the goal is to find a backbone tree such that the average delay of communication between any endpoints is minimized. The delay is semantically defined in terms of costs associated to every link in the network.

A routing load or cost $C(e_{i,j})$ in a tree $T = (V, E)$ associated with an edge $e_{i,j} \in E$ is equal to the product of the weight $w_{i,j}$ and the cardinalities $Deg(X_i)$ and $Deg(X_j)$ between the endpoints X_i and X_j respectively. Equation 5.4 describes this metric.

$$C(e_{i,j}) = Deg(X_i)Deg(X_j)w_{i,j} \tag{5.4}$$

We are interested in developing a clustering algorithm that aims to partition subsets of highly correlated variables, but we are also aiming for clusters that are not too small or too big with respect of the full domain of variables χ. Thus we detect and delete the arcs whose associated cost is maximal. At first sight, it is not clear why we took this decision. However, the reason is founded in terms of the cardinality of the variables. In other words we guide the search for "inconsistent" edges for those regions of the tree that are highly connected.

Opposing to the previous objective functions, we want to minimize equa-

tion 5.3. This is because we want to minimize the damage of removing an edge whose weight might be of a relevant magnitude. Thus, in cases in which two edges $e_{i,j}$ and $e_{h,l}$ have the same cardinality product $(Deg(X_i) * Deg(X_j)) = (Deg(X_h) * Deg(X_l))$ then we choose to delete that arc whose weight $w \in \{w_{i,j}, w_{h,l}\}$ is minimal.

Edges that connect leaf variables with the rest of the tree have higher probability of being discriminated since its own cardinality is low. Equation 5.5 describe the objective function for the MCST procedure and Figure 5.5 presents its algorithm.

$$W^G = \sum_{i=1}^{k} \sum_{X_h, X_j \in V_i} C(e_{h,j}) \tag{5.5}$$

The algorithm is almost identical to the SEMST algorithm, the only difference is in the calculation of costs (lines 1-4) and the selection of edges $e_{r,s}$ whose associated $cost_{r,s}$ is maximal. The pruning and construction of clusters behave exactly in the same fashion which was seen in the previous section. The complexity of the MCST algorithm primarily relies in the calculation of the degree for each variable, having that we count with an adjacency matrix for each edge $e_{r,s} \in EMAST$ then it takes at most $O(n * 2d)$ operations to complete the calculation for each individual cost (whereas d is the maximal cardinality of a given variable $X_{\in\chi}$).

Evidently, this algorithm as well as the SEMST algorithm do not directly optimize a specific concrete function. This unsupervised partitioning is of course not optimal if we try to evaluate it in terms of a stricter target such as equation 5.2. On the other hand, this approach might respond significantly better than the SEMST algorithm (in fact, this asseveration is empirically proved in the experimental results in section EXPERIMENT.SECTION).

Minimum Cost Spanning Tree Algorithm (MCSTA)

Input: $MAST = (\chi, EMAST)$, W^{MAST}, $k \in \mathbf{Z}^+$
Output: $C = \{C_1, C_2, \ldots, C_k\}$

1: FOR $h = 1$ to $EMAST$
2: $e_i = (X_r, X_s)$
3: $cost_{r,s} = |X_r| * |X_s| * w_{r,s}$
4: $Costs \Leftarrow cost_{r,s}$
5: FOR $i = 1$ to $(k-1)$
6: $I = (I \cup e_{r,s})$ iff $(X_r, X_s) \notin I \wedge e_{r,s}$ has a $cost_{r,s} = \underset{cost \in Costs}{\text{argmax}}\ (cost)$
7: FOR $j = 1$ to I
8: $e_{r,s} = (X_r, X_s)$
9: $EMAST = (EMAST - e_{r,s})$
10: **IF** $i = 1$
11: $G_1 \Leftarrow R_{X_r}$
12: $G_2 \Leftarrow R_{X_s}$
13: $G \Leftarrow G_1$, $G \Leftarrow C_2$
14: **ElSE**
15: **FOR** $h = 1$ to $|G|$
16: **IF** $e_h \in EG_h$
17: $G_1 \Leftarrow R_{X_r}$
18: $G_2 \Leftarrow R_{X_s}$
19: $G = (G - G_h)$
20: $G = (G \cup G_1)$, $G = (G \cup G_2)$
21: FOR $i = 1$ to $|G|$
22: $C_i = V_i$ having that $G_i = (V_i, E_i)$

Figure 5.5: The Minimum Cost Spanning Tree Algorithm

5.1.4 Zahn Euclidean Minimum Spanning Tree Algorithm

The previous MAST-based clustering attempts used a truly heuristic rule to partition a given domain. However, the rule they use to select candidate edges for deletion is greedy and offers no guaranties in global terms. For this reason, it is interesting to evaluate an algorithm which not only focus in a

given arc $e_{r,s}$; but also verifies the set of neighboring arcs in N_r and N_s with respect of its endpoints X_r and X_s.

In this case we can visualize that the MCST and SEMST algorithms perform a greedy blind search over the dependency graph in order to select inconsistencies. On the other hand, the Zahn's Euclidean Minimum Spanning Tree (ZEMST) algorithm firstly introduced by Zahn [70] takes into account not only a given arc but its relevance among its neighbors. This clustering process is based in the comparison between a weight $w_{r,s}$ and the arithmetic measures such as the means ($\bar{w}_{N_r}$, $\bar{w}_{N_s}$) and the standard deviations (σ_{N_r} , σ_{N_s}) of neighboring weights (w_{N_r}, w_{N_s}).

These neighboring weights are related to the edges in the subgraphs belonging to the neighbor hods $N_r = (V_{N_r}, E_{N_r})$ and $N_s = (V_{N_s}, E_{N_s})$ were V_{N_r} and V_{N_s} are the set of variables; and E_{N_r}, E_{N_s} are the set of edges expanding from the variables X_r and X_s (not counting the direction that includes the edge (X_r, X_s)).

Let us define the methods that are necessary for detecting "inconsistent" edges in the ZEMST algorithm. Firstly, the mean $\bar{w}_{N_r}$ of a neighborhood N_r can be defined as:

$$\bar{w}_{N_r} = \frac{1}{|E_{N_r}|} \sum_{(X_i,X_j)\in E_{N_r}} w_{i,j} \tag{5.6}$$

Secondly, the standard deviation σ_{N_r} of neighborhood N_r is equivalent to:

$$\sigma_{N_r} = (\frac{1}{|E_{N_r}|} \sum_{(X_i,X_j)\in E_{N_r}} (w_{i,j} - \bar{w}_{N_r}))^{\frac{1}{2}} \tag{5.7}$$

Before any calculation takes place we need to define the the size of the subgraph N_r (neighborhood). For this reason we need a parameter d that is the diameter of N_r. In other words. This neighborhood can be easily obtained for any edge by performing a classic bread first search with depth

d over the Maximum Spanning Tree. We proceed to examine the ZEMST algorithm in Figure 5.6.

Zhan's Euclidean Minimum Spanning Tree Algorithm (ZEMSTA)

Input: $MAST = (\chi, EMAST)$, W^{MAST}, $d \in \mathbf{Z}^+$, $c, f \in \Re$
Output: $C = \{C_1, C_2, \ldots, C_k\}$

1: FOR $i = 1$ to $|EMAST|$
2: $e_{r,s} = (X_r, X_s)$
3: Construct $N_r = (V_{N_r}, E_{N_r})$, $N_s = (V_{N_s}, E_{N_s})$ of depth d.
4: Calculate $\bar{w}_{N_r}$ and $\bar{w}_{N_s}$ using equation 5.6.
5: Calculate σ_{N_r} and σ_{N_s} using equation 5.7.
6: **IF** $(w_{r,s} < (\bar{w}_{N_r} - \sigma_{N_r}))$ **OR** $(w_{r,s} < (\bar{w}_{N_s} - \sigma_{N_s}))$
7: **IF** $|G| = 0$
8: $G_1 \Leftarrow R_{X_r}$, $G_2 \Leftarrow R_{X_s}$
9: $G \Leftarrow G_1$, $G \Leftarrow G_2$
10: **ElSE**
11: **FOR** $j = 1$ to $|G|$
12: **IF** $e_{r,s} \in E_j$
13: $G_1 \Leftarrow R_{X_r}$, $G_2 \Leftarrow R_{X_s}$
14: $G = (G - G_j)$, $G = (G \cup G_1)$, $G = (G \cup G_2)$
15: **ELSE IF** $\frac{w_{r,s}}{max(c*\bar{w}_{Nr}, c*\bar{w}_{N_2})}$
16: **IF** $|G| = 0$
17: $G_1 \Leftarrow R_{X_r}$, $G_2 \Leftarrow R_{X_s}$
18: $G \Leftarrow G_1$, $G \Leftarrow G_2$
19: **ElSE**
20: **FOR** $j = 1$ to $|G|$
21: **IF** $e_{r,s} \in E_j$
22: $G_1 \Leftarrow R_{X_r}$, $G_2 \Leftarrow R_{X_s}$
23: $G = (G - G_j)$, $G = (G \cup G_1)$, $G = (G \cup G_2)$
24: FOR $i = 1$ to $|G|$
25: $C_i = V_i$ having that $G_i = (V_i, E_i)$

Figure 5.6: The Zahn's Euclidean Minimum Spanning Tree Algorithm

The algorithm receives the Maximum Spanning Tree $MAST$, the set of weights W^{MAST}, the depth d and two threshold parameters c and f. First, it

is important to note that all the calculations are always taken from the full $MAST$. We obtain the neighborhoods N_r and N_s by a search of depth d (line 3). Then, for each edge $e_{r,s} \in EMAST$ we calculate the means $\bar{w}_{N_r}$, $\bar{w}_{N_s}$ for the neighborhoods N_r, N_s and the standard deviations σ_{N_r} and σ_{N_s} (lines 4-5). Now in order to decide if an edge $e_{r,s}$ is inconsistent we perform two tests. First we check if the weight $w_{r,s}$ is significantly smaller than any of the means minus its standard deviations (line 6), and then we just assembly the auxiliary graphs as in the previous algorithms (lines 7-14). The second test (line 15) removes those outliers that are far away from the mean (targeting those edges whose weight is significantly higher than the mean), and if this equality holds then the auxiliary graphs are redefined (lines 16-23). Finally, every variable $X \in V_i$ contained in an auxiliary subgraph $G_i = V_i, E_i$ is mapped to a cluster C_i (lines 24-25).

The complexity of the ZEMST algorithm is more elaborate than its predecessors, since it has to perform a search of at most $d-1$ adjacent variables; thus, the most significant part of the algorithm runs (lines 3-5) in a time with a lower boundary in $O(n * d)$ and a worst case scenario in $O(n^2)$ whenever $d \approx n$. The organization of the auxiliary subgraphs in lines and could be handled in a time $O(|G|c)$ if appropriate data structures are used (c is the highest number of variables in a subgraph).

We can see from Figure 5.6(lines 6 and 15) that the ZEMST algorithm basically removes all the edges that are not close to its mean. However, it does not only removes those edges presenting low weights but also those ones that are proportionally higher to the highest of the two neighboring means. Indeed the algorithm targets a different objective function; that is to keep all the edges that are arithmetically closer to the mean.

5.2 From Complex Networks to the Star Discovery Algorithm

The last MST-based clustering approaches perform clustering by executing simple pruning rules. We can intuitively realize that, as the rules for partitioning become more complex then the final clustering seems to be of a better quality. Thus, the search for "inconsistent" edges is directed to "convenient" zones (regions in which by cutting edges we can obtain clusters with a fair number of variables). However, we are interested in finding a competitive and robust algorithm that could be offered as another option besides ACA in order to find a set of clusters. Even though, we have not seen convincing results by using a MAST for performing clustering that does not represent that such dependency graph is not valuable.

On the other hand, we aim in this section for a robust MAST partitioning algorithm that is mostly founded in graph theory operations, and motivated by the theory of complex networks [71, 72]. Complex networks has its roots in the study of interactions between a group of different entities. Therefore, it is an abstraction and a study of topologies, interactions, densities of clusters of nodes, dynamics of networks, etc.

Complex networks seek for "hidden" phenomenas that are behind a given network that can be described in terms of graph theory. One of the fields of complex networks is the study of scale-free networks [73] that are nothing but graphs that initially start with a given number of variables and edges and then whenever they grow through the pass of time they seem to conserve the same topology (i.e. initial dense zones of connected components remain with a similar connectivity while other regions remain sparse). A scale-free network is a graph whose nodes have a degree distribution that follows a

power law distribution $P_i(k) \sim k^{-\gamma}$, whereas $P_i(k)$ is the probability that a given node connect other k nodes (γ is an empirical threshold).

This field of study is new and it is currently widely studied. There exist many examples of scale-free networks such as interaction between groups of persons in social networks or epidemics. However, there are other examples of networks that are believed to be scale-free networks such as the world wide web since some patterns in its growth suggest a geometrical extension.

Focusing back in our search for a good clustering algorithm we may ask to ourselves: How is that scale-free networks relate to attribute clustering? Indeed, if we can reduce our problem to a graph partitioning problem, and then into a tree clustering problem; we could follow the intuition relying some empirical properties of scale-free networks. It has been reported in [71] the hypothesis that in real world cases such as the world wide web, (see Figure5.7) there may exist a correlation between pairs of nodes that hold a high degree of connectivity. However, these asseverations are still in study, since there is not nearly complete data regarding the exact topology for every domain in the WWW. The Opte project [74] is one of the efforts to describe related structures and patterns on the net. Once we acquire knowledge regarding connectivity and faithful patterns; then the discovered topology could be analyzed.

Scale-free networks was the premise that motivated the previous work of Ninna Paivinen in [75] in order to formulate a simple but acceptable clustering algorithm based in scale-free networks. The clustering algorithm is based on the modification of Prim's algorithm [76] for MST construction. In that approach a tree is iteratively constructed with the difference that, whenever a node that is already in the MST reaches a certain degree then it starts to absorb more new nodes than its counterparts. the later is possible

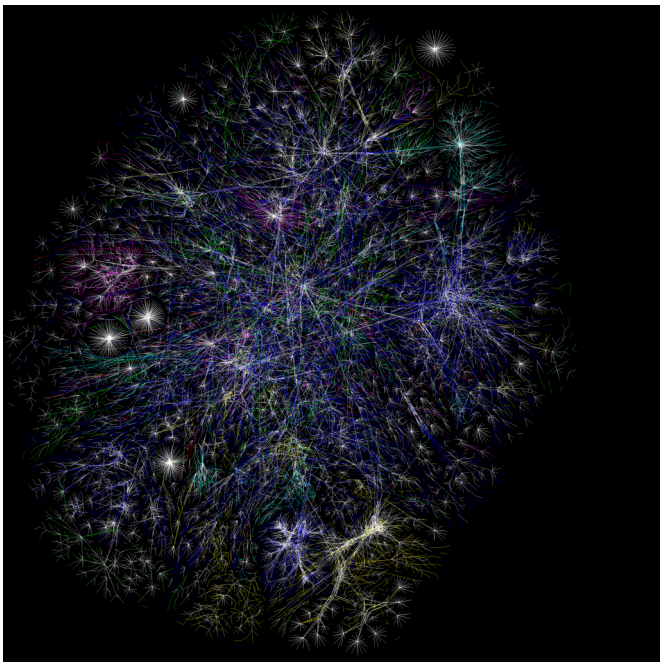


Figure 5.7: A Shallow description on several domains in the WWW. Each color represents a given domain, i.e. Yellow= $\{.net, cn, au\}$, Blue= $\{.net, .ca, .us\}$.

by affecting the weight of that node versus the rest of the domain. Then, a final process searches for nodes with a high degree, and then pulls these center nodes as well as its branches into several clusters.

The novel idea of using the SFMST algorithm (over a conventional k-medoid algorithm) is to direct the search of regions of highly correlated variables. Such grouping of variables may provide a clearer picture of the domain. Therefore, those k partitions are identical to those regions in the SFMST that present a high density of variables. However, the definition of a region with a high density of variables is heuristic (depending in some predefined rules). Biasing the placement of nodes in a tree might intuitively reduce the problem of selecting centers of performing bisections in a given dependency graph. However, the process of biasing might be unnecessary, since

a Maximum Spanning Tree can provide enough features for the selection of centers or modes.

5.2.1 The Star Discovery algorithm

In this section the Star Discovery algorithm (SD) is introduced. This approach follows the same clustering principle as the previous work. We select take a Maximum Spanning Tree and then we search for candidate modes or centers depending on some constraints. Once a candidate node is selected as a true mode it is stored in a cluster with its adjacent and leaf nodes. The process is repeated until no more variables remain unassigned into clusters (exhaustive clustering).

Before we analyze the approach in more detail, we will review a key concept in which the SD algorithm is based upon. Guiding the search for centers by only examining the topology is probably not a good idea, since we completely ignore the weights. Actually, from the previous algorithms we can see that they based the clustering in a very simplistic combined search involving topology and weights. However, by examining only isolated edges or neighborhoods for clustering trees is probably not a good idea. We may exploit further features from this dependency graph.

A sound and clear approach is to look for subgraphs from the MAST that could reveal information about the "nature" of the domain. One abstraction of our technique is to look for spanning stars as subgraphs contained in the MAST.

A spanning star [77] is a spanning tree of m nodes over the initial Maximum Spanning Tree. A spanning star $S = (VS, ES)$ of $m = |S|$ nodes has one center $o \in VS$ with a degree equals to $m - 1$. All the other nodes in the spanning star have a degree of one.

In fact, the idea of the k-modes clustering algorithm is indirectly based on the same idea of discovering stars. In the k-modes, the process starts by selecting a random set of modes and then all the other nodes are associated to a given star such that the weight is maximal. Later, a new set of modes are recalculated and the process repeats once more until no new stars are found. Actually, the k-modes is the optimal algorithm in order to find stars since it has complete information among all pairwise interactions in the domain. However, detecting the set of k-stars whose global weight is maximal from a complete graph K_n requires a high number of computations.

Without loss of generality over the whole domain, we can input the Maximum Dependency graph and perform the discovering of stars on its topology. Any graph that contains n variables can have up to n different stars. We aim to detect a set of clusters that were originated from a set of stars ζ such that equation 5.8 is maximized.

$$W^{\zeta} = \sum_{S_i \in \zeta} \Big(\sum_{X_j \in A} (w_{i,j}) + \sum_{X_j \in A, X_h \in L} (w_{j,h}) \Big) \tag{5.8}$$

Notice that we extended the notion of a star S to include the leaf nodes L connecting adjacent nodes A to a center o. The later is done because during experimentation we found that leaf nodes have a higher correlation to the center of its adjacent node than to any other center in any other star.

The SD algorithm as well as the previous algorithms receives a Maximum Spanning Tree $MAST$ and the set of weights W^{MAST}. Initially, it forms n different stars $S_i \in \zeta$, by selecting each variable X_i as center o_i, and attaching its adjacent nodes A and leaf nodes L (lines 2-6). Once that all possible stars ζ have been constructed we proceed to the partitioning task (lines 7-9). Finally, the algorithm selects the set of variables SV_j to become the new clus-

Star Discovery Algorithm (SDA)

Input: $MAST = (\chi, EMAST)$, W^{MAST}
Output: $C = \{C_1, C_2, \ldots, C_k\}$

1: FOR $i = 1$ to $|\chi|$
2: $o_i^{'} = X_i$
3: $A_i^{'} \Leftarrow X_j$ iff $(X_i, X_j) \in EMAST$
4: $L_i^{'} \Leftarrow X_h$ iff $(X_h, X_j) \in EMAST \wedge X_j \in A$
5: $S_i^{'} = (o_i \cup A_i^{aux} \cup L_i^{aux})$
6: $\zeta^{'} = S_i^{'}$
7: WHILE $V^{aux} \neq \emptyset$
8: $C_i = SV_j^{'}$ iff $w^{S_j^{'}} = \underset{w \in W^{'}}{\text{argmax}}(w)$
9: $V^{aux} = (V^{aux} - VS_j^{'})$

Figure 5.8: The Star Discovery Algorithm.

ter C_i iff the weight w^{S_j} of the star $S_j = (SV_j, SE_j)$ is maximal(line 8). The process is repeated until all variables are assigned to clusters (condition in line 7). Assuming that there are n variables, the complexity of the algorithm is dominated in its first phase (star construction in lines 1-6) since it takes $O(n^3)$ operations to search for all the adjacent nodes and leaves.

The star discovery algorithm always provide solutions that are deterministic. Thus, if the algorithm runs for several times it will always give the same solution. On the other hand, the star discovery might not offer results that are better in quality than the ones given from the k-modes algorithm (k-modes has complete information about the domain). However, ACA based in k-modes could grant better solutions in many cases but it has the risks of falling in local optima. Chapter EXPERIMENT.CHAPTER, present several comparisons made to evaluate each of the algorithms presented in this chapter.

Chapter 6

Recovering Bayesian Networks from Clusters

In previous chapters, we have discussed several properties that are important in order to gain reduced knowledge from a given domain. We have seen in Chapter 4 that the use of dependency graphs such as the Maximum Spanning tree simplify the nature of the domain (i.e. a MAST can approximate with an acceptable). In Chapter 5, we presented and studied the process of dividing a full domain into clusters of highly correlated attributes. These subsets of highly dependent variables are in fact valuable, since they reveal compact features from the domain study (i.e. isolating genes that are responsible for detonating upheavals related to leukemia [35]).

Now, it is relevant to study the relevance between many components in order to learn the structure of Bayesian networks. We present the hypothesis that clustering χ in to several subsets C is in fact the same problem of identifying "regions" from a complete Bayesian network.

This chapter introduces a class of BN structural learning algorithms. This class is based in the "divide and conquer" principle. We have the objective of dividing the instance of a structure learning problem into a more tractable scenario. First, we introduce the motivation for divide and conquer problem

solving. Then, we present related work in the field and finally, we formulate the Overlapping Structure Learning algorithm for recovering the structure of Bayes nets.

6.1 Adopting a Divide and Conquer Paradigm

The computational divide and conquer strategy sections an instance of a problem into two, three or more smaller and clearer instances. Each new instance of the problem encodes an easier but important feature of the original problem. We have that whenever these smaller instances can be solved readily, the solution of the original problem can be achieved by combining these small components. Therefore, the divide and conquer approach is a "top-down" class of solutions. Furthermore, if the smaller solutions of a problem are still large enough they may be divided into some tractable instances.

Well known algorithms (commonly used by programmers and computer scientists) employing the divide and conquer paradigm are: Binary search, merge-sort, quick-sort, Strassen's matrix multiplication algorithm between many others. On the other hand it is also important to know when not to apply this paradigm. It has been stated in literature [78] that this approach is not useful when:

- A problem of size n is divided into several instances each almost of size n.

- An instance of a problem with size n is divided into almost n instances of size $\frac{n}{c}$ where c is a constant.

6.2 Learning Bayesian networks from Distributed Data

This section provides related work that motivated the formulation of our further learning algorithm. Before we continue with the discussion regarding overlapping partitions, lets recall two important constraints that must hold in order to learn proper Bayesian networks: The Markov Assumption and the Faithfulness assumption.

- Markov assumption: Some variable X_i is conditionally independent of all the non descendants and not non parents given $Pa(X_j)$.
- Faithful assumption: If two variables X_i and X_j are conditional independent given the separator set S, then there is no edge in the BN.

Logically, the Markov assumption is the central idea behind the factorization of joint probability distributions into n different probabilities. On the other hand, the faithful assumption ensures that if an edge exists in the BN is because they are real causals, such that no subset $S \subset \chi$ can make them conditional independent.

Two key factors motivate the study of learning Bayesian networks from multiple subsets of variables. First, suppose we want to learn the BN from several datasets. These datasets could have overlapping or common variables; therefore, at one point one might result interested in merging one or more datasets that are situated in different locations. Social datasets for example have common variables and are used probably for different purposes. A knowledge engineer might suddenly become interested in learning a BN in each dataset at first, but there can also be the case that he is interested in learning the full BN.

On the other hand, the second factor that motivates this approach is the reduction of complexity for learning BN by a set of partitions. The previous section presented the divide and conquer paradigm and we may formulate learning BNs from data in the same fashion. In concrete, learning a BN can be formulated as learning overlapping partitions of its variables. Then combining and merging the resulting networks in order to convey to the full network.

One of the first attempts to solve the combination problem was shown in [79] by Danks. Such work is a constrained based learning algorithm. In his work, he proposed a learning algorithm that combines components by following heuristics that are similar to the PC algorithm [6]. Specifically, that work introduced two important rules that aim to deal with the combination of two smaller BNs $BN^A = (G^A, P^A)$ (such that $G^A = (A, EA)$ having a set A of variables and a set EA of edges) and $BN^B = (G^B, P^B)$ such that ($G^B = (B, EB)$ having a set B of variables and a set EB of edges) into a complete Bayesian network $BN^\chi = (\chi, E)$ where $\chi = A \cup B$ and $M = A \cap B$. Figure 6.1 graphically presents those relevant sets of variables for this discussion.

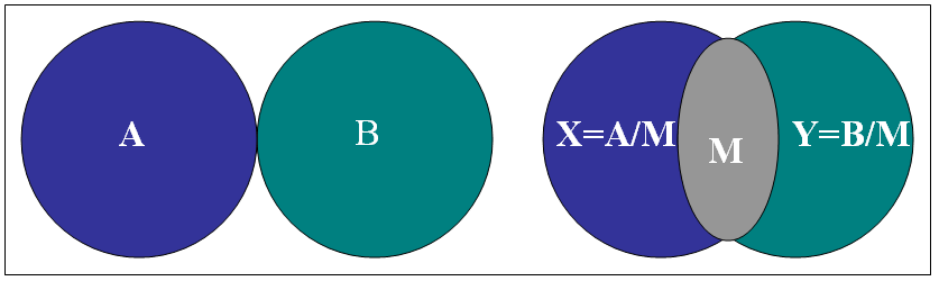


Figure 6.1: The overlapping (A and B) sets of variables.

The first rule states that if two variables X_i and X_z are not adjacent in either G^A or G^B then they must not be adjacent in G^χ. Thus, there is no

dependency between them. The second rule says that for all $X_i \in X = A/M$, $X_j \in Y = B/M$ and $X_h \in M$ if there is a directed path from X_i to X_h in the G^A that involve only variables in X; and X_h and X_j are not adjacent in G^B, then X_i and X_j are not adjacent in G^χ.

The implications of the previous rules ensure that in any combination algorithm only those relevant edges will be considered for G^χ. If we consider a constraint learning algorithm, in which we start with the full undirected graph, then these rules can be understood as edge removal heuristics. Thus, rule one discards those edges that do not show causality in any subset. While the second rule removes those edges that are conditional independent to X_j. If X_h and X_j are not adjacent it is because there exist another set of variables that makes them d-separated; therefore, any ancestor of X_h that is only contained in G^A can not be adjacent in the final graph ensuring this way the Markov and faithful assumptions.

Finally, we present the structure learning using prior results (SLPR) algorithm. It is basically, an instantiation of the PC algorithm with a focus in independence tests around M, and those variables in either X or Y that have links to the elements of M. Following we present a general description of each of the steps in the SLPR algorithm.

I. Start with the complete undirected graph and remove those edges that do not exist in either G^A or G^B (applying rule the first deletion rule). For all the elements in M delete those edges that appear in only one graph (this indicates that the variables are conditionally independent given a d-separation set in the other graph). Make undirected those edges in M that are in conflict (i.e. X_i,$X_j \in M$ and $X_i \rightarrow X_j$ in G^A and $X_i \leftarrow in X_j$ in G^B).

II. For all $X_i \in X = A/M$, $X_j \in Y = B/M$ and $X_h \in M$ if there is a

directed path from X_i to X_h in the G^A that involve only variables in X; and X_h and X_j are not adjacent in G^B, then delete $X_i \rightarrow X_j$ in G^χ (this is a direct application of the second rule as described above).

III. Test for conditional independence for each edge that connects variables between X and Y by considering a d-separation set S of size k. This rule is exactly applied as in the PC algorithm (Performing conditional tests by incrementing the size of the d-separation set k).

IV. Produce v-structures. For each triple of variables X_i, X_j and X_h, such that there exist the edges $X_i - -X_j$ and $X_j - -X_h$ but X_i and X_h are not adjacent; then orient $X_i - -X_j - -X_h$ for $X_i \rightarrow X_j \leftarrow X_h$ iff X_j is not in the separator set between X_i and X_h.

V. Test for independence all edges that connect any element in M from either X or Y. Indeed, in the combination phase, we want to assure the faithful assumption, such that no pair of variables are connected given a d-separation set. Thus, since we are merging BN^A and BN^B there could be elements in B (A) that make independent those adjacent variables in $M \cup A (M \cup B)$.

VI. For all edges in G^χ are undirected. Then we apply (in order) the following two rules:

For each triple of variables X_i, X_j and X_h, such that there exist the edges $X_i - -X_j$ and $X_j - -X_h$ but X_i and X_h are not adjacent; then orient $X_i - -X_j - -X_h$ for $X_i \rightarrow X_j \leftarrow X_h$ iff X_j is not in the separator set between X_i and X_h.

Iteratively, apply the following sub-rules until all edges in G^χ have been oriented:

If $X_i \rightarrow X_j - -X_h$ and X_i, X_h are not adjacent, then orient $X_j - -X_h$ as $X_j \rightarrow X_h$.

Avoid cycles. If there is a directed path from X_i to X_j, and X_i, X_j are adjacent, then orient $X_i - -X_j$ as $X_i \rightarrow X_j$.

Initially, the SLPR algorithm originally takes the d-separation sets obtained by earlier learning algorithms which obtained BN^A and BN^B. Then it performs a set of tests and heuristic rules in order to for the final BN. This algorithm has shown to have a polynomial complexity equivalent to the cardinalities of the sets $O(|X||Y| + |A||X| + |B||Y| + |M|(|A| + |B|))$ in the worst case scenario.

6.3 The Overlapping Structure learning Algorithm

The SLPR algorithm seems to be a good option to learn two Bayesian networks if they have overlapping variables. However, there is an inconvenient of using the SLPR algorithm for further learning. First, the SLPR algorithm has shown to be efficient with only a pair of variables; thus, it might be useful to extend it in order to learn and combine a set of k clusters. Another issue is that, the SLPR algorithm makes full of prior d-separation sets. The later implies that the algorithm in fact, will work only if the previous subsets of variables were learned by a another constraint based method. This consideration narrows the potential of the algorithm in order to experiment with different types of structure learning algorithms.

In this section we introduce the Overlapping Structure learning (OSL) algorithm. The OSL algorithm aims to overtake those limitations offered

by the SLPR algorithm. We are interested in this case in learning the full structure of the BN but from more than two subsets of variables. One may think that the SLPR algorithm can be used to perform such task. However, the problem arises whenever more than two clusters of variables overlaps.

In short terms, the OSL algorithm receives a dataset with a complete set of variables χ and then it performs attribute clustering in order to detect disjoint components. Once that the components are identified, they are expanded and merged into the full BN in a smart way. Figure6.2 depicts the OSL algorithm.

Overlapping Structure Learning Algorithm (OSLA)

Input: $D = \{a_{1,l}, \cdots, a_{n,l}\}$ and parameters PAR
Output: A BN $BN^{\chi} = (\chi, E)$

1: Load the data D.
2: Detect a dependency graph H.
3: Partition H into overlapping clusters OC.
4: Form a Cluster Graph $CG = (U, F)$ from OC.
5: **WHILE** $i = 1 < |U|$
6: Learn a Bayesian knot K_{U_i} for cluster OC_i.
7: $K \leftarrow K_{U_i}$.
8: Combine all $K_{U_i}, K_{U_j} \in K$.

Figure 6.2: The Overlapping Structure Learning algorithm.

The OSL algorithm receives the Data D having a sample size $l = \Omega$ (each attribute a_i is mapped to a discrete random variable X_j, with a domain) and a set of parameters PAR (these parameters are necessary for the clustering and combination parts that will be described in the next two subsections). First, the algorithm loads D and calculates a dependency graph H for this domain (lines 1-2). Next, we proceed to divide H into smaller overlapping clusters and we form a Cluster Graph ($G = (U, F)$ with the set of clusters

U and the set of edges F) using the methods that are described in the next section (lines 3-4). Then, for each cluster $U_i \in U$ we learn its BN Knot K_{U_i} and we store it in the superset K(lines 5-7). Finally, we combine all the Knots $K_{U_i}, K_{U_j} \in K$ (following the topology dictated by the Cluster Graph CG) and we form the full Bayesian network BN^{χ} (line 8).

Logically, if we opted for using a divide and conquer strategy for solving the instance of a problem (in this case learning Bayesian networks); then we may require fast and reliable partitioning and merging processes. For this reason we develop the OSL algorithm in a straightforward fashion. We just need to recover those local regions inherent in the full BN and then just add those links that connect single components between each other. Thus, the OSL algorithm is an hybrid process since its complexity depends in several exchangeable subroutines (division and combination of components). Therefore, the complexity of the algorithm will be dominated by the specific instances of the algorithms that are used to handle each of its steps (we can input any division process in line 3 such as those techniques introduced in the previous chapter).

Eventually, we may see that if the number of components is relatively small compared to the full set of variables in χ then the algorithm can be competitive in terms of time only if the combination phase can be covered in a fair amount of computational time. The next subsections will explain in detail the subroutines that are necessary in order to produce overlapping clusters, generate the Bayesian knots and how to combine the Bayesian network components into a final Bayesian network BN^{χ}.

Overlapping Clustering

So far we have been discussing the big picture of the problem. In this subsection we will discuss in detail how is it possible to partition a given dependency graph H into several components that are in fact overlapping. During the course of this investigation we found that there are two major approaches towards the generation of overlapping clusters. We can either run an specialized overlapping clustering algorithm or we can turn a disjoint set of clusters into overlapping ones.

The overlapping partitioning cluster (OPC) algorithm introduced by Chen in [80] performs non-exhaustive clustering by assigning an element to the nearest clusters according to two constraints: The first one is to maximize the distance between centers in different Clusters. Secondly, and probably more importantly it maximizes the number of variables in each Cluster.

The OPC algorithm seems to fit our needs; However, there exist some flaws in its formulation. Fist, we need to perform exhaustive clustering in order to detect the desired number of reduced components (It is not of much use to discover clusters that contain only one variable). Conclusively, we may like to discover those local components that really reveal a high correlation between its true contents (We may loose local correlations if we move the centers far away from each other from a global perspective). Having this in mind we proceeded to develop our own overlapping clustering algorithm in order to meet our specific demands. We are interested in discovering a set of disjoint clusters that hold true dependencies between its contents and then we proceed to expand each cluster.

We introduce the Overlapping Expansion (OE) algorithm. The OE algorithm relies in the class of fuzzy clustering approaches and is constructed based in similar grounds than [81]. Suppose we have the complete graph over

the set of points or variables that compose the clusters. In this way we can formulate this problem in terms of euclidean distance. The basic idea behind this algorithm is to expand the clusters C in order to include the nearest neighbors that are contained in other clusters (producing the set of overlapping clusters $OC = \{OC_i | i = 1, \ldots, n\}$). The number of nearest neighbors that will be include in every cluster C_i is given by equation 6.1.

$$f^{C_i}(\epsilon) = \lceil |C_i| * \epsilon \rceil \tag{6.1}$$

where $f^{C_i} : \Re^+ \mapsto Z^+$, $|C_i|$ is the cardinality of the cluster C_i and ϵ is an empirical threshold that aims to control the number of nodes to be added to the current cluster ($0 \geq \epsilon \leq 0$). Equation 6.1 defines a fuzzy number of variables or points that will be included to C_i equal to the ceil function of a ϵ percentage of the number of variables in C_i. Its fuzzy function is graphically expressed in Figure 6.3.

Notice from Figure 6.3 that we can automatically control the number of nodes to be added to each cluster by just varying ϵ. All clusters with be expanded in the same proportion. Intuitively, the assignation of variables (in the case of random variables) works by sorting those weights connecting all variables in C_i to outliers. Then, we choose the variables $OV \notin C_i (|OV| = f^{C_i}(\epsilon))$ with the highest relationship (minimal distance) to a single element in C_i.

$f^{C_i}(\epsilon)$ is bijective since two elements in OV can point to the same variable in C_i and two or more variables in C_i can output to the same variable in OV. In this way we ensure to recover only the most correlated variables that were not initially included in any of the clusters. Figure 6.4 shows the OE algorithm.

From Figure6.4 we can see that it inputs a set of clusters and then maps

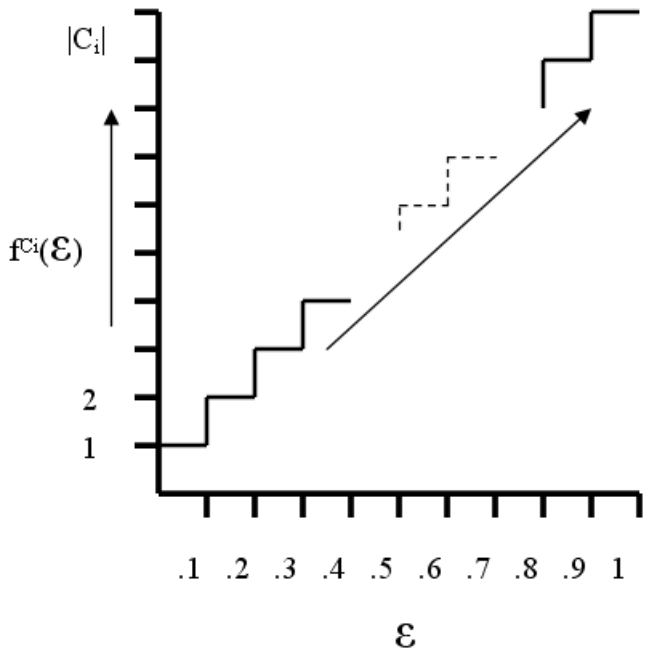


Figure 6.3: The function $f^{C_i}(\epsilon)$ calculates the number of nearest variables (belonging to other clusters) that will be included into C_i to produce the overlapping cluster OC_i.

every $C_i \in C$ into the new $OC_i \in OC$ (line 1). Then for all clusters, we expand $OC_i \in OC$ by detecting its nearest neighbors OV (lines 3-8). The expansion rule to follow is to include those true nearest variables to each cluster C_i (line 5). The complexity in this case is governed by the search of the set OV and it is in the order of $O(ns)$ where s is the maximal cardinality of any given cluster C_i. The whole process of this algorithm is explained in Figure 6.5 by using a small example involving four clusters and 21 variables.

The example in Figure 6.5 with a fixed parameter $\epsilon = 0.1$ and set of clusters $C_1 = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}\}$, $C_2 = \{X_{11}, X_{20}, X_{14}\}$, $C_3 = \{X_{12}, X_{13}, X_{15}, X_{16}, X_{17}\}$ and $C_4 = \{X_{18}, X_{19}, X_{21}\}$ works in the fol-

Overlapping Expansion Algorithm (OEA)

Input: $C = \{C_1, C_2, \ldots, C_k\}$, $\epsilon \in (0, 1]$, χ, W^G
Output: $OC = \{OC_1, OC_2, \ldots, OC_k\}$

1: $OC = C$
2: $OV = \emptyset$
3: **FOR** $i = 1$ to k
4: **FOR** $h = 1$ to $\lceil |C_i| * \epsilon \rceil$
5: $X_h = \underset{w_{h,j} \in W}{\text{argmax}}(w_{h,j})$ iff $(X_h \in C_i) \wedge (X_j \neq OV) \wedge (X_j \in (\chi - C_i))$
6: $OV = OV \cup \{X_j\}$
7: $OC_i = OC_i \cup OV$
8: $OV = \emptyset$

Figure 6.4: The Overlapping Expansion Algorithm

lowing way: a)Initial setup. b)C_1 is the candidate cluster to be expanded. c)$X_{21} \in C_4$ is the nearest variable to $X_8 \in C_4$. d)C_2 is the new candidate cluster to be expanded. e)$X_{19} \in C_4$ is the nearest variable to $X_{11} \in C_4$. f)C_3 is the cluster to be expanded. g)$X_4 \in C_1$ is the nearest variable to $X_{15} \in C_3$. h)Finally, C_4 is the candidate cluster to be expanded (Although C_4 is already connected to C_1 the expansion rule states that we are interested in verifying its nearest variable). i)$X_8 \in C_1$ is the nearest variable to $X_{21} \in C_4$.

We can apply any of the algorithms studied in Chapter 5 to find a set of disjoint clusters; and then convert them into overlapping groups by using the OE algorithm. However, there is a drawback related to this process. If we only consider the expansion formulated in terms of the nearest neighbors, we have the possibility of finding isolated regions of clusters. For example, suppose we have four clusters $C_1, C_2, C_3 and C_4$ (with the same cardinality —C1—=—C2—=—C3—=—C4—=10) and we set ϵ to 0.1. Therefore, we aim to expand each cluster with one external variable. Then, after running

the OE algorithm we find that C_1 is connected to C_2 as well as C_3 is linked to C_4. We can see that there is no connection between the regions $C_1 \cup C_2$ and $C_3 \cup C_4$.

The objective now is to generate a general structure that can help us to integrate the small instances into the full BN. Therefore, in order to deal with the later issue, we provide a new algorithm that ensures connectivity and complete reachability between every pair of clusters in C. The resulting general structure is denominated the Cluster Graph CG. The Cluster Graph is based in the Cluster Tree presented by Friedman et al. in [53]. Specifically, A Cluster Graph is a pair $CG = (U, F)$ where U is a set of overlapping clusters and F indicates the set of edges $(C_i, C_j) \in F$ representing adjacent clusters C_i and C_j.

We developed the Cluster Graph (CG) algorithm to discover the cluster topology between a given set of clusters. This process actually do not augment the components in a fuzzy proportion. It verifies that the full set of clusters are reachable. In a negative case, it locates the isolated regions, and then merges them into a full CG by adding only the nearest variable to every isolated region.

In other words, we find the union R for all the overlapping clusters produced by the OE algorithm. If more than one region is found ($R = \{R_1, \ldots, R_l\}$), then we start expanding every region $R_i = \{C_1, C_1, \ldots, C_m\}$ with the nearest variable $X_j \in \chi - R_i$. X_j will be stored in the cluster(s) that has a variable X_i such that $w_{i,j}$ is maximal (nearest neighbors). Figure 6.6 introduces the Cluster Graph algorithm.

The CG algorithm receives the overlapping clusters OV, a complete set of variables χ and the set of all weights W^{χ}. Then, we form a region R by searching all reachable nodes in U (lines 3-5). Next, if the cardinality of R

is smaller than $|\chi|$ then we obtain the nearest neighbor X_j (lines 6-7) and we add it to the nearest cluster(s) included in R (lines 8-10). Finally, we construct the set of edges F (lines 11-14). The complexity of this algorithm is dominated in lines 3 to 7 and it is in the order of $O(st)$ where s is the maximum number of variables in a region and t is the maximum number of variables outside R (logically, the extensive search is done between these subsets of χ). The rest of the steps (lines 8-10 and lines 11-14) are just every case scenarios in the order of $O(k)$ and $O(k^2)$ that aim to add variables and edges to U and F respectively.

6.3.1 Learning Bayesian Network Knots

Once that we have ensured total connectivity between all the variables, we learn every cluster separately. Each cluster has a subset of domain variables. The final procedure is to learn the directed acyclic graph (DAG) for each cluster by using any available learning technique. The output is a set of knots $\{K_1, \cdots, K_i, \cdots, K_m | K_i = (U_i, E_i, P_i)\}$. Each knot has a DAG structure (U_i, E_i) (with the set of variables U_i and the set E_i of edges) and a probability distribution P_i associated to Y_i. The procedure of recovering Knot structures is shown in Figure 6.7.

The complexity depends on the selected learning technique. For example, if the PC algorithm [6] is used, the complexity of line 2 is in the order of $O(mq^s)$, where s is the largest number of parents for a node, q is the largest cardinality of a cluster. In general $q \ll n$, the complexity of learning knots is trivial in comparison with learning the complete network.

6.3.2 Combination Phase

The final procedure of the OSL algorithm is the combination of the different knots. We follow a simplistic procedure in order to recover the full Bayesian network BN^{χ}. The procedure we employ is based in some parts of the SLPR algorithm. Basically, we avoid to conduct expensive conditional independence tests and we aim to just use the following combination rules:

I. Start with the global skeleton provided by the Cluster Graph $CG = (U, F)$ and the structures that were learned in the knots K. For all the arcs residing inside overlapping areas $M = \{(U_1, U_2), \ldots, (U_r, U_s)\}$ delete those edges that appear in only one knot. Make undirected those edges in the overlapping zones that are in conflict (i.e. X_i,$X_j \in M$ and $X_i \rightarrow X_j$ in K_{U_i} and $X_i \leftarrow in X_j$ in K_{U_j}).

II. For all edges in G^{χ} are undirected. Then we apply the following two rules:

    If $X_i \rightarrow X_j - -X_h$ and X_i, X_h are not adjacent, then orient $X_j - -X_h$ as $X_j \rightarrow X_h$.

    Avoid cycles. If there is a directed path from X_i to X_j, and X_i, X_j are adjacent, then orient $X_i - -X_j$ as $X_i \rightarrow X_j$.

Given that we have the Cluster Graph CG, we just need to respect its topology, we keep the edges that are learned in the knots and we only solve conflicts in the overlapping areas M of CG. We can have an efficient combination by only using these rules because we assume that the clusters contain subsets of highly correlated variables from χ. The conditional independences were tested beforehand in the learning knots phase. This process differs with the SLPR algorithm in which we already have the true correlated subsets of

variables. On the other hand, the SLPR algorithm needs to perform more testing due to the fact that the algorithm integrates different datasets that were never tested from a global point of view. In our case the clusters proceed from a dependency graph that encoded a true topology of the domain of study; in this way, we can have more reliable clusters.

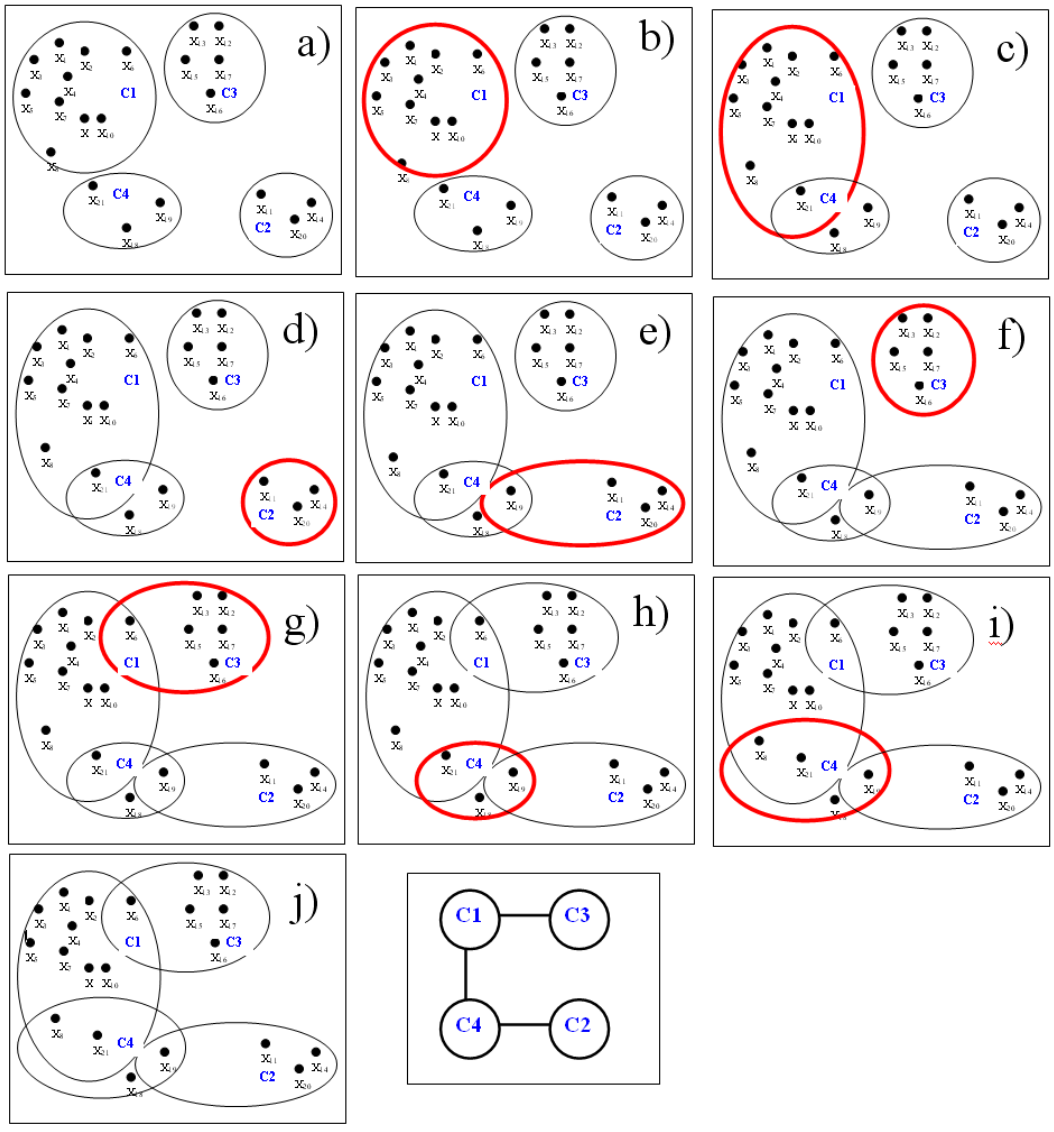


Figure 6.5: An example of the OE algorithm with $\epsilon = 0.1$. In this case, every cluster will be expanded by only one variable according to Equation 6.1.

Cluster Graph Algorithm (CGA)

Input: $OC = \{OC_1, OC_2, \ldots, OC_k\}$, χ, W^G
Output: $CG = (U, F)$

1: $U = OC$, $F = \emptyset$
2: **WHILE** $R \neq \chi$
3: **FOR** $h = 1$ to k
4: **IF** $(R \cap U_h) \neq \emptyset$ **OR** $|R| = 0$
5: $R = R \cup U_h$
6: **IF** $|R| < |\chi|$
7: $X_j = \underset{w_{i,j} \in W}{\text{argmax}}(w_{i,j})$ iff $(X_i \in R) \wedge (X_j \in (\chi - R))$
8: **FOR** $r = 1$ to k
9: **IF** $X_i \in U_r$ **AND** $U_r \subseteq R$
10: $U_r = U_r \cup X_j$
11: **FOR** $i = 1$ to $k - 1$
12: **FOR** $j = i + 1$ to k
13: **IF** $(U_i \cap U_j) \neq \emptyset$
14: $F = F \cup \{(U_i, U_j)\}$

Figure 6.6: The Cluster Graph Algorithm

Learning Bayesian knots Algorithm (CGA)

Input: $U = \{U_1, U_2, \cdots, U_k\}$, Data $D = \{a_{1,l}, \cdots, a_{n,l}\}$
Output: $K = \{K_1, K_2, \cdots, K_k\}$

1: Load the data D and $C = \{C_1, C_2, \cdots, C_k\}$
2: Build $K = \{K_1, K_2, \cdots, K_k\}$ by applying a structure learning technique.
3: LReturn $K = \{K_1, K_2, \cdots, K_k\}$

Figure 6.7: A Bayesian network is learned for each overlapping cluster.

Chapter 7

Experimental Results

In this Chapter we will investigate the quality of the results of several algorithms proposed in this thesis. Initially, I present a set of experiments regarding attribute clustering; in this group of experiments I compare the k-modes (ACA) algorithm versus the other partitioning approaches such as the SEMST, ZEMST, CEMST and SD algorithms. Following, several BN structural experiments are presented in order to test the efficiency of the OSL algorithm. We compare the OSL algorithm against some traditional learning algorithms that recover the full structure by accessing the complete set of variables χ.

I developed a KD system written in Java in order to produce the clusters of attributes and the Bayesian networks for all the methods presented in this work. The system makes use of MySQL to store results and the Hugin [82] system to visualize the generated Bayesian networks. Besides this, and given that our OSL algorithm is dependent of a primary learning algorithm (in order to learn the small clusters), we used the structure learning algorithms employed in the Causal Explorer package [83]. Between the Bayesian networks involved in these experiments[1] we have:

[1]The DAG of these networks can be found at the end of this thesis in Appendix A.

- The Alarm BN ($|\chi| = 37$) [58].

- The Barley BN ($|\chi| = 48$). The BARLEY network is a real world application. It assists in the production of beer from Danish barley [84].

- The HeparII BN ($|\chi| = 70$). The HEPAR II network helps to diagnose some types of liver malfunctions [85]. The nodes cirrhosis, chronic hepatitis, hepatic steatosis and toxic hepatitis present a high degree.

- The Hailfinder BN ($|\chi| = 56$). This Bayesian network was built with the purpose to forecast summer hail in the northeastern part of Colorado [86]. One of its variables (Scenario) has a high out degree

- The Pathfinder BN ($|\chi| = 109$).The PATHFINDER network provides knowledge domain for the diagnosis of lymphatic-node sicknesses [87]. Notice in Appendix A the highest out degree of the Fault node.

7.1 Attribute Clustering Experiments

In this section we will review several features regarding attribute clustering. Firstly, we will investigate the reliability of the AC algorithm (k-modes) to find local and global maximums. Next, we proceed to compare ACA versus all the other attribute clustering methods shown in Chapter 5 in terms of quality of the results.

7.1.1 Reliability Tests for the Attribute Clustering Algorithm

Firstly, we will focus in testing the reliability of the AC algorithm. Initially, ACA was tested in [39] in order to test its ability to reveal true clusters of variables (by using a synthetic dataset). Nevertheless, such experiment is not convincing regarding reliability. Thus, the synthetic dataset which was employed was very well divided. Furthermore, we would like to test the performance of this method in other real and more complex datasets. In fact, since the formulation of the k-modes algorithm is greedy, there is the risk of falling into local optima. Therefore, in order to test the susceptibility of ACA to fall in local optima, we feed it in each domain with all the possible $\binom{n}{2}$ combinations of variables.

For this experiment, we used datasets that were generated by Hugin [82] setting a sample size of 10,000 from the following BNs: Alarm, Barley, HeparII, Hailfinder and Pathfinder. Figure 7.1 presents a table with the different modes, weights that were found in each dataset.

We found that indeed ACA does fall in local optima. For example, in the Alarm domain, it was interesting to see that ACA converges into an optimal value of 6.13 with modes VentAlv and HR. However, it falls into two local optima in 5.91 and 4.06 having VentAlv and LVEDVolume (VentAlv, Shunt) as modes. In the optimal result, the size of the clusters is about $\frac{n}{2}$. In the local optima, the cluster containing LVEDVolume (Shunt) is relatively small (10 variables). Clearly the small cluster is the isolated because of the suboptimal value. Actually, whenever LVEDVolume (Shunt) is selected as a mode, then no improvement is made. This mode dominates its neighborhood. The previous analysis is a straightforward example of techniques based solely in iterative greedy search. For the other domains the situation becomes more

dramatic. For Barley, Hailfinder, Pathfinder and HeparII we found 17, 91, 117 and 130 local optima respectively (however, the for these domains the differences between the local results and the optimums are very close).

7.1.2 Comparing the Clustering Quality

Even though, every attribute clustering algorithm used a different objective function, it is interesting to evaluate the quality of the final results. In fact, a sound estimate in order to evaluate the goodness of a cluster of attributes is given in [39] by the following Equation:

$$W^C = \sum_{i=1}^{k} \sum_{j=1}^{|C_i - o_i|} w_{j,o_i} \tag{7.1}$$

In other words, we are concerned to calculate the degree of dependency between the center or "mode" o_i of each cluster C_i against its elements. Then, we get the global weight of the clustering by adding each measure from each cluster. Equation 7.1 is in fact the objective function of the AC algorithm. For this reason we may expect that ACA is the optimal answer (whenever it does not converge in local optima) for this experiment.

For this experiment we used artificially generated datasets from the following BNs: Alarm, Barley, HeparII, Hailfinder and Pathfinder. In every case, we considered and used several sample sizes in order to test the response of the algorithms to different statistics. We supplied different datasets with various sample sizes ranging from 4000 to 10000 in intervals of 2000. Figures 7.2, 7.3, 7.4, 7.5, and 7.6 present all the results in terms of global weight and sample size for each domain. Notice that the X-Axis is the supplied sample size and the Y-Axis is the global weight W^C. For all algorithms we set $k = 4$. In the case of ZEMST we set the depth to 15, c to 1 and f to 0.1. In all cases

for the expansion we chose a minimal setting for $\epsilon = 0.1$. We empirically found that those settings produced the best results.

Actually, it is easy to conclude just by appreciation that the SD algorithm actually performs better than the other clustering algorithms. Indeed, sometimes the SD appear higher than ACA. This does not mean that SD is more optimal than ACA. It means that, ACA was kept in a local optima (this is a theoretical claim since I used the best value for ACA in each plot). We can also conclude that the most unstable clustering algorithm is the $ZEMST$ algorithm since it performs very poorly in some domains such as Alarm and Barley. Conclusively, we can learn that the MAST is actually useful whenever a sound clustering method is applied to section it.

7.2 Structural Experiments for Bayesian networks

In order to test the efficiency of the OSL algorithm, we performed a set of structural experiments. We learned several Bayesian networks by using our method and other learning techniques. We employed the MDL score [88, 89] to test the quality of the resulting networks. The MDL score is a well known metric for testing the structural quality of Bayesian network. The MDL metric is based in the principle that minimizes the encoding length of the model, and the encoding length of the data given the model. It is important to notice that the MDL principle will obtain an optimal trade off between the accuracy and the complexity from a given model to represent the data [88].

In every case, we considered and used several sample sizes in order to test the response of the algorithms to different statistics. The values for the re-

duced statistics oscillated from multiples of $|\chi|$ times $d = \{6, 9, 12, 15, 18, 21\}$. The values for the sufficient statistics were normally from 2000 to 10000 in intervals of 2000.

7.2.1 Results on the Alarm network

We conducted this experiment with the aim of testing the performance of the OSL algorithm to learn the full BN. We employed the 4 different structure learning algorithms in Causal Explorer to learn the knots: The PC algorithm [6], the Three-Phase Dependency Analysis TPDA [90], the Sparse Candidate SC [53] and the MaxMin Hill Climbing MMHC [29] algorithms.

Before we continue with our experiment, it is relevant to recall the previous results that were offered in [29] regarding the performance among these learning algorithms. We used datasets generated in Hugin for the Alarm BN with several sample sizes. Then, we used each of the learning algorithms to generate the full BN. We can see from Figure 7.7 that the MMHC algorithm is the best option for learning Bayesian networks. However, we will investigate the performance of the OSL algorithm by using each of these four algorithms.

In order to investigate the response of the OSL algorithm we chose the two best clustering algorithms: The SD and the ACA approaches. k was set to 4 in the case of ACA in order to produce higher quality of results. We know from experience that, as k tends to be small the clusters also tend to improve. Figures 7.8, 7.9, 7.10 and 7.11 present the structure learning results for the PC, TPDA, SC and MMHC algorithms.

Interestingly enough, we can see that SD again performs better in general (even though, in some sample sizes i.e. SC from $\Omega = 6000$ to 1000, ACA performed slightly better due to the higher dimensions of its clusters).

In fact, the OSL algorithm (with the help of the SD clustering approach) performs significantly better than the PC, TPDA, and SC algorithms. We chose the same learning algorithm in every case in order to make this comparison admissible. We can conclude that the OSL algorithm performs better in the constraint based algorithms because it restricts the sets of variables to be target for conditional independence sets. We know that the smaller the number of variables the more reliable the tests are.

On the other hand, for the SC algorithm the reason of the big improvement is also clear, we are avoiding noisy variables by restricting only those highly correlated variables as parents for each variable in every cluster. In all cases the isolation of clusters inhibits false or redundant arcs between different sets of variables (this indirect pruning in fact help to avoid extra edges).

Finally, in the case of the MMHC algorithm the OSL algorithm is almost as competitive. However, further extensions to our algorithm could help to converge into the best results (i.e. using some conditional tests in order to ensure faithfulness between variables which around the neighborhood of the overlapping sets). To conclude our experimentation in the Alarm domain we recorded the elapsed times for all these learning algorithms. Notice that I just include the learning time in the knots, since the computation of the complete graph is still very expensive. the clustering methods, overlapping extensions and combination phases spend no more than 2 seconds in any instance of the OSL algorithm. It is extremely important to devise a way to approximate the calculation of the MAST; in order to provide competitive time. Figures 7.12 and 7.13 present the timing results for the OSL using the SD and the ACA algorithms respectively[2].

[2]I will present a set of comprehensive experiments during the oral examination. For technical reasons I was not able to include all results in this document.

Domain	Modes	W
	VentAlv, LVEDVolume	5.91554
Alarm	**VentAlv, HR**	**6.13758**
	VentAlv, Shunt	4.06153
	hcv_anti, vh_amn	54.7639
	alcoholism, hcv_anti	54.7384
HeparII	**hcv_anti, hospital**	**54.8288**
	hcv_anti, surgery	54.8257
	hcv_anti, gallstones	54.7482
	hcv_anti, amylase	54.6537
	hcv_anti, injections	54.8231
	jordn, komm	2.16854
	jordn, nedbarea	2.16832
Barley	**ksort, jordn**	**3.33046**
	jordn, antplnt	2.90914
	jordn, forfrugt	2.13446
	jordn, pesticid	2.16885
	nprot, ngodn	2.55821
	jordn, nopt	2.31252
	N0_7muVerMo, ScenRelAMCIN	33.7037
	CombVerMo, ScenRelAMCIN	34.0806
	ScenRelAMCIN, Date	33.7405
Hailfinder	**ScenRelAMCIN, Scenario**	**34.2835**
	WindFieldMt, ScenRelAMCIN	33.5981
	InsInMt, ScenRelAMCIN	33.8798
	ScenRelAMCIN, CompPlFcst	34.0279
	ScenRelAMCIN, CombMoisture	33.7297
	Fault, F72	92.2702
Pathfinder	**F97, F72**	**94.3609**
	F72, F53	91.8937
	F72, F6	91.8503
	F72, F7	91.8259
	F72, F56	91.9068
	F72, F8	91.843

Figure 7.1: Different convergent modes for each domains. In every domain, we denote the optimal result in bold. The rest of the combinations of modes are local optima.

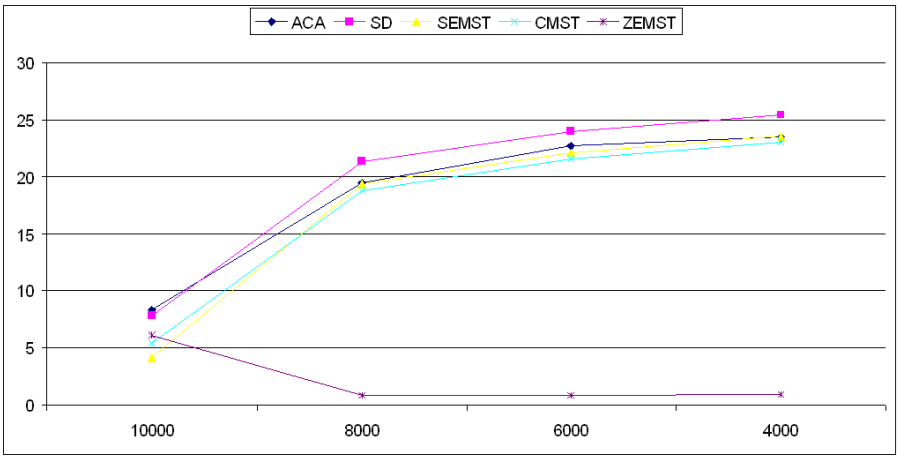


Figure 7.2: Clustering results for the Alarm Domain.

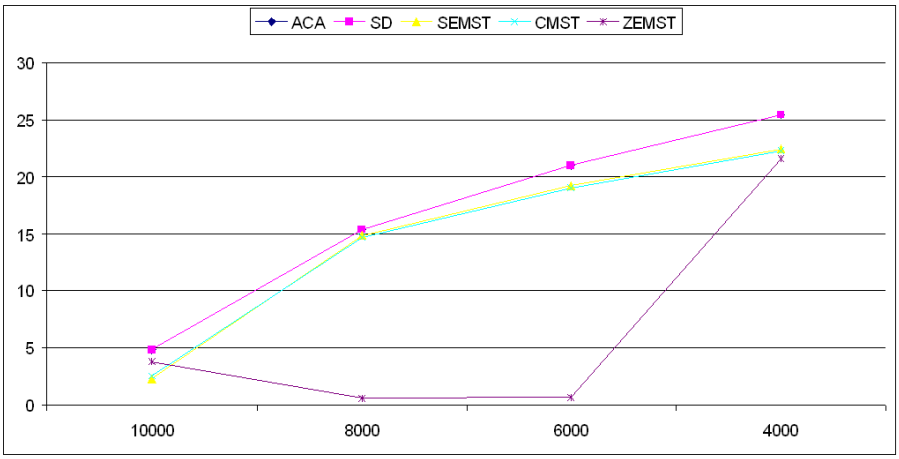


Figure 7.3: Clustering results for the Barley Domain.

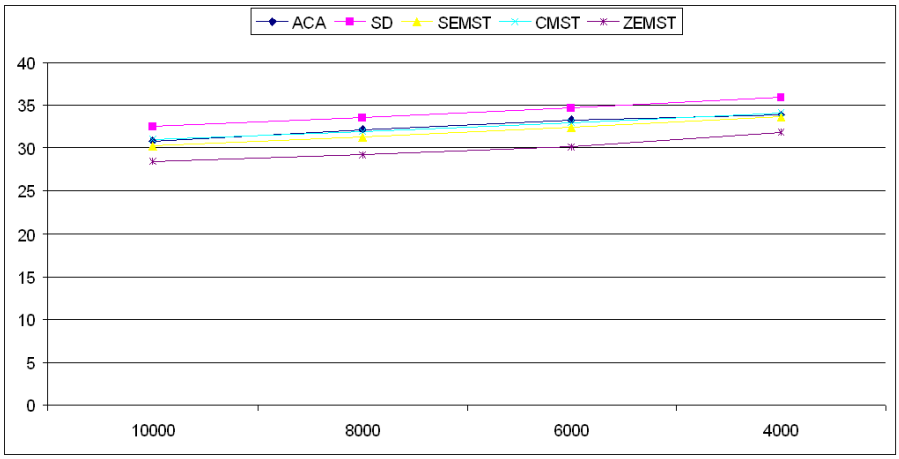


Figure 7.4: Clustering results for the Hailfinder Domain.

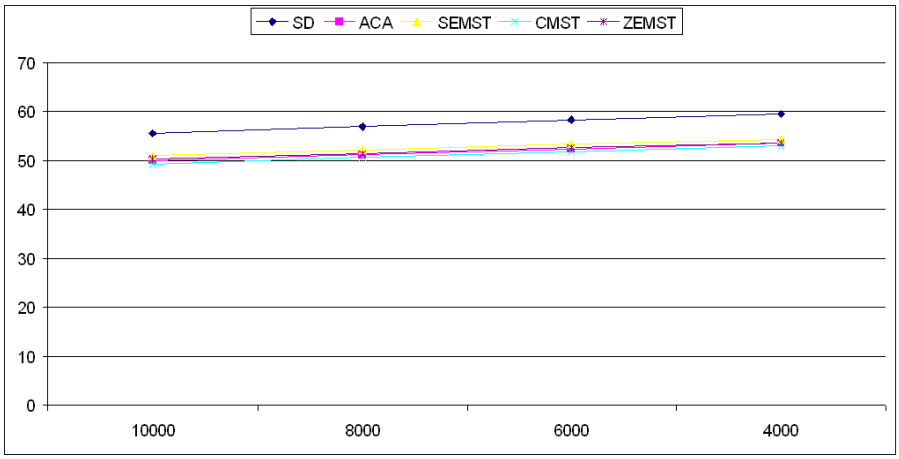


Figure 7.5: Clustering results for the HeparII Domain.

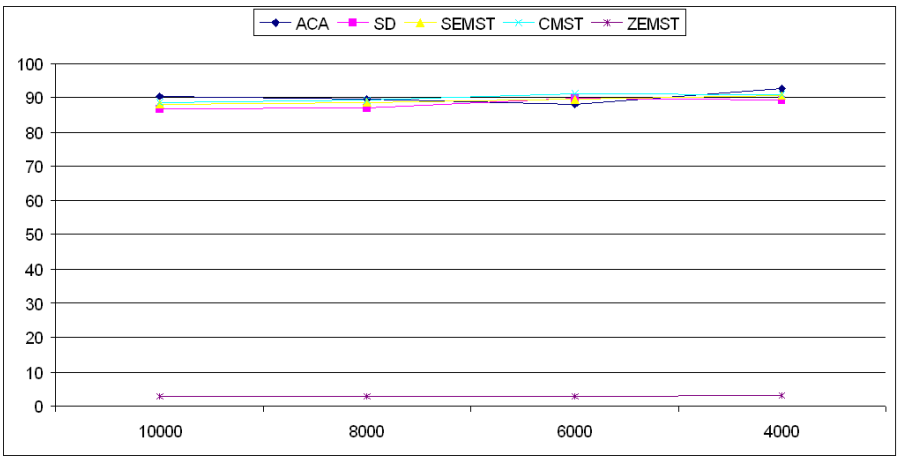


Figure 7.6: Clustering results for the Pathfinder Domain.

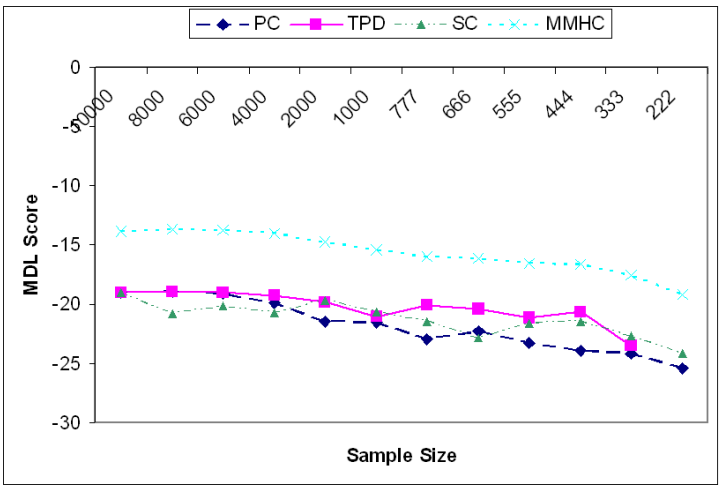


Figure 7.7: Structure learning results for the full networks in the Alarm domain. MMHC is clearly the best algorithm.

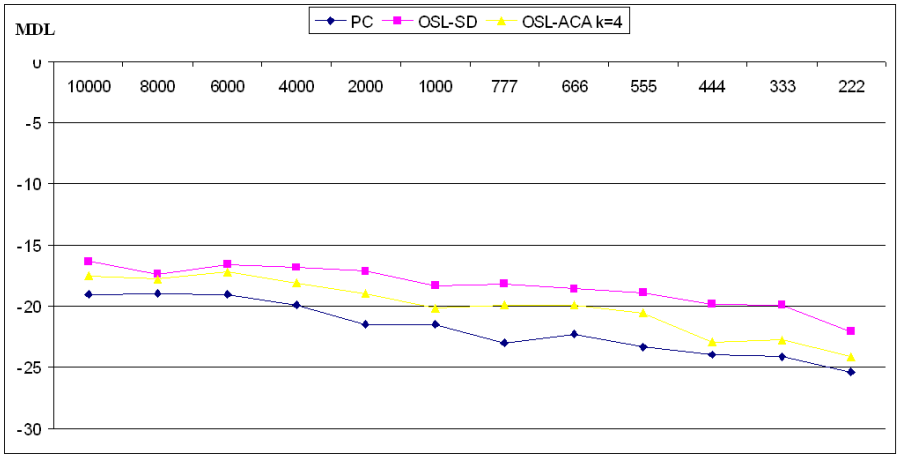


Figure 7.8: Structure learning results produced by applying the PC algorithm.

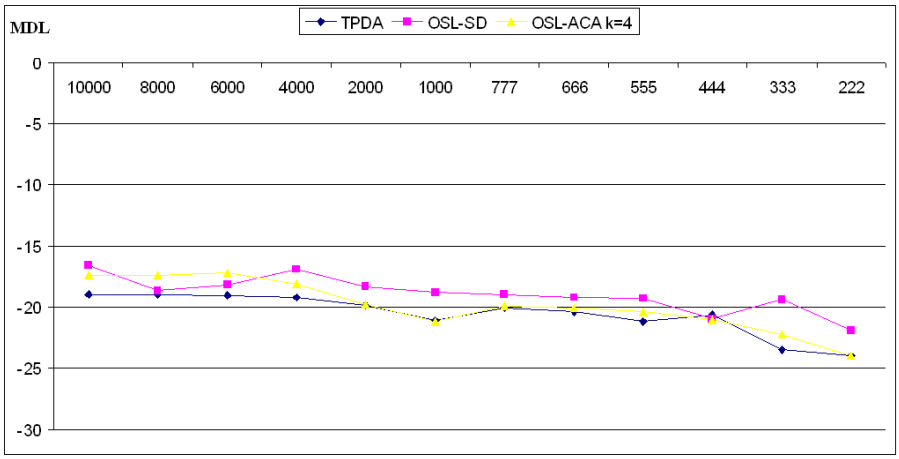


Figure 7.9: Structure learning results produced by applying the TPDA algorithm.

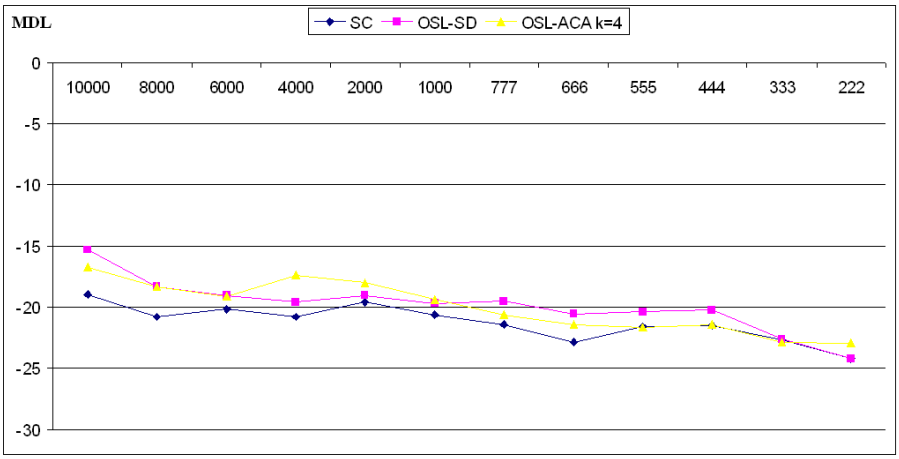


Figure 7.10: Structure learning results produced by applying the SC algorithm.

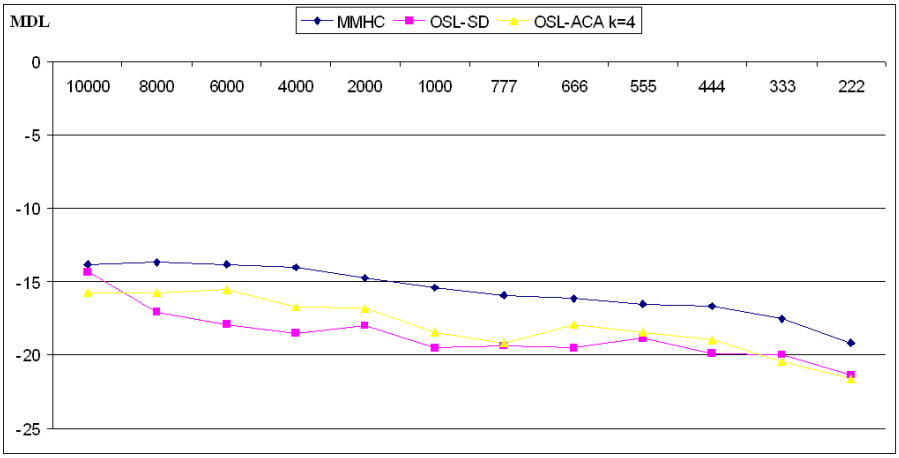


Figure 7.11: Structure learning results produced by applying the MMHC algorithm.

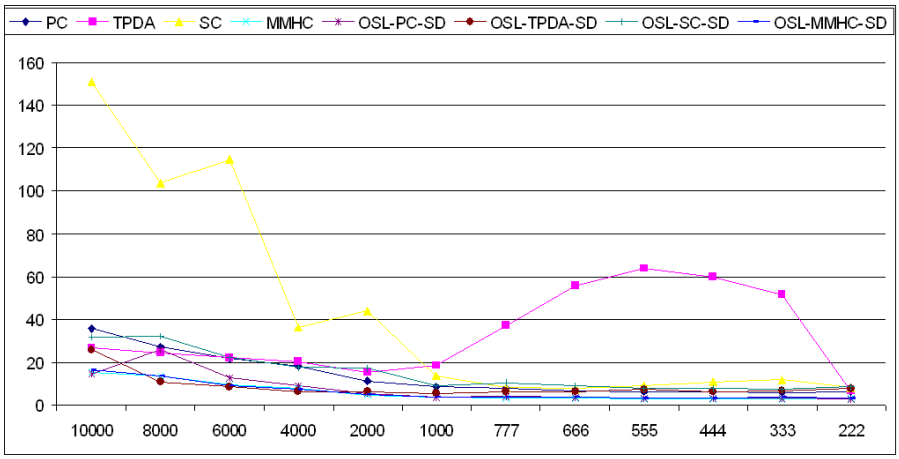


Figure 7.12: Time results in all instances of the OSL-SD algorithm.

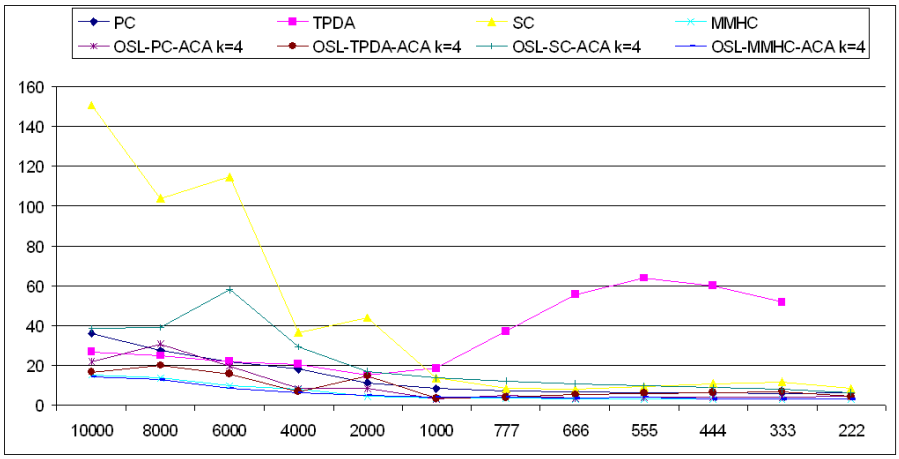


Figure 7.13: Time results in all instances of the OSL-ACA $k = 4$ algorithm.

Chapter 8

Conclusion

This final Chapter reveals some of the thoughts of the author regarding the proposed approaches for attribute clustering and learning Bayesian networks. Firstly, the reader will have the opportunity to visualize several tracks of future work founded in the current research. Finally, some conclusions regarding this investigations are provided.

8.0.2 Future Developments

Further work is in several levels of this research. In the case of attribute clustering we have seen that the AC algorithm provide the best solutions. However, the only drawback regarding this method is the risk of falling into local optima. We may propose some further method that could help to avoid this problem (i.e. some search technique that may aim to calculate the global clustering before the elements are assigned to clusters in the third phase of the algorithm, in this way we may detect local optima at a certain deepth). Moreover, the attribute clustering methods are well suited to calculate and detect the local dependencies among entities in several biological datasets.

In the case of the learning algorithm a number of improvements can be done. Initially, we have that in most cases the sum of the sectioning time,

learning time of the knots and combination time is faster than the learning time (by other algorithm) of the full domain. However, the only issue regarding time in the OSL algorithm is the calculation of the complete graph. As we know, the idea of generating a complete graph is to be able to built the MAST from the dataset. The time that is necessary to generate a complete graph is quadratic in terms of the attributes and lineal in terms of the number of observations. Even though, that construction of the complete graph is in polynomial time, it is surely an expensive and many times not affordable computation. For this reason, we may opt for developing a new method in order to approximate the MAST. The MAST approximation is primarily founded by Moore et al. in [60].

This new method works in the following way: First, we start with a random tree, then we perform tests over all pairs of variables in the tree that are not connected. We calculate the weight between them; next, we detect the path that makes then reachable in the tree. If the weight of the recent considered edge, is higher than a weight attached to an arc connecting this pair of variables in the tree, then we substitute that arc in the tree for the new one (deleting the edge in the tree and adding the new one).

Concerning the learning algorithm itself, the combination phase could be improved. We may use a set of conditional tests (or a small score Hill climbing approach) in order to solve the conflicts in those arcs residing inside the overlapping regions of the cluster graph. Currently, we employ a set of rules that aim at forming the full network. We defend the fact that most clusters contain all the relevant v-structures and most relevant parent sets of variables. However, there is the risk of loosing some arcs between variables connecting elements in the overlaps. It is for this reason that a further step could validate the faithfulness of the overlapping regions. On the other

hand, we could apply an iterative score-based method that adds, removes or reverses edges between the overlapping variables and its neighbors. In any case, this further steps will help to finally improve the full BN (probably ending up with a result at least as good as the BNs produced by the MMHC algorithm).

We also could test the Bayesian network knots for representing reduced knowledge from regions of full network. We can learn the parameters of the knots (possibly with the EM algorithm). Then, use a metric (such as the KL divergence) in order to test the deviation between the probabilities in the simplified network and the full setting. Finally, we could propose some use of the knots for evidence propagation. We know that propagating evidence in large domain is a very hard task. For this reason, we may restrict the propagation to only those elements inside the clusters. Then, we could propagate that evidence with the rest of the network through the overlapping variables (passing evidence from clusters to clusters of variables). Conclusively, some other clustering algorithms or sectioning methods could be used in order to generate the subsets of variables for learning.

It is important to notice that the author is currently involved in the calculation of more results for the learning algorithm. More structural results will be provided during the presentation in the final examination. In the case that the reader is interested in the new comprehensive set of experiments.

8.0.3 Final Conclusion

In general, the present work extended the notion of attribute clustering for learning Bayesian networks from Data. We reduced the problem of learning Bayesian networks from a full set of variables χ into the learning of local components of variables (knots). Moreover, the work presented in this thesis

involves in detail two aspects: The study of attribute clustering and a new approach for learning the structure of Bayesian networks.

The major contributions of this work are:

- The development of a KD system that performs attribute clustering and structure learning of Bayesian networks.

- We proposed several MAST based clustering algorithms for attribute clustering (in the past, only the k-modes algorithm had been used for this purpose).

- We introduced two new clustering algorithms: The first one is denominated the Star Discovery algorithm (SD). The SD algorithm is based in the intuition behind scale free networks (the degree of a variable in a network also denotes its relevance to the inner domain, highly connected variables are likely to be the most dependent and relevant features in a given domain). The second clustering algorithm named the Cost Maximum Spanning Tree algorithm (CMST) was a natural extension to the notion of tree partitioning; instead of deleting the k edges according to their lowest weights, we considered the costs (which involved cardinality of the variables). These two algorithms can actually be used for traditional clustering or attribute clustering.

- A new BN structure learning algorithm called the Overlapping Structure Learning algorithm (OSL) was introduced. Experimental results show that the OSL algorithm is a competitive approach. Furthermore, the OSL algorithm proved to be more efficient than learning full domains with other traditional methods (such as the PC, TPDA and SC algorithms).

In terms of attribute clustering, we found that the traditional k-modes approach the AC algorithm provides in some cases, the best clustering. However, it falls into local optima. On the other hand, the SD algorithm was close to ACA in terms of quality but performed faster. Moreover, the SD algorithm does not have the risk of falling into local optimums since it is a deterministic approach. In regards of the clustering methods, we found that there is a correlation between the degree of complexity and the quality of the results. Robust and more strict clustering algorithms find better clusters in longer elapsed times. Thus, the ranking (in order) from best to worst clustering algorithms is: ACA, SD, ZEMST, CMST, SEMST. In terms of elapsed times the list repeats backwards from the fastest to the slowest (SEMST, CMST, ZEMST, SD, ACA).

The OSL algorithm provided a new framework for learning Bayesian networks. Its process can be easily described in these steps: Identification of a dependency graph, clustering of the dependency graph, expansion of disjoint clusters, construction of a cluster graph, learning of local components or knots and combination of knots. A set of experiments in the Alarm BN show that the OSL algorithm is better than two constrained learning algorithms(PC and TPDA) and one score based algorithm (SC). For the constrained learning algorithms, the OSL restricts the set of conditioning sets in all variables (thus, a smaller number of variables is more reliable for testing independences). In the case of the Sparse Candidate algorithm. The OSL approach restricts the "parent" variables to be only the ones contained in a single cluster (since the variables that are located in clusters are the most mutual correlated, the restriction is done over only the relevant sets of variables).

Conclusively, we can state that the problem of structure learning of

Bayesian networks, and finding clusters of variables are truly related; and in fact, complementary. Good clusterings encapsulate groups of truly correlated variables. Those correlated sets are adjacent variables in the full BN. The potential of the current work deserves further study and development such that it could provide optimal results by using further auxiliary combination steps (i.e. conditional independence tests or score-based structure search over those reduced set of overlapping variables connecting clusters). The idea of recovering reduced knowledge in the form of local components is indeed valuable for describing the full domain.

Bibliography

[1] F. V. Jensen, *An introduction to Bayesian networks.* New York: Springer, 1996.

[2] F. V. Jensen, *Bayesian Networks and Decision Graphs (Information Science and Statistics).* Springer, 2002.

[3] I. N. Friedman, M. Linial and D. Pe'er, "Using bayesian network to analyze expression data," *Journal of Computational Biology*, no. 7, pp. 601–620, 2000.

[4] R. e. a. Marot, B. G.; Holthausen, "Using bayesian belief networks to evaluate fish and wildlife population viability," *Forest Ecology and Management*, vol. 1-3, no. 153, pp. 29–42, 2000.

[5] R. E. Neapolitan, *Learning Bayesian Networks.* Prentice Hall, 2004.

[6] G. Spirtes, P.; Glymour and R. Scheines, *Causation, Prediction and Search.* New York: Springer-Verlag, 1993.

[7] R. e. a. Cheng, J.; Greiner, "Learning bayesian networks from data: an information-theory based approach," *The Artificial Intelligence Journal*, no. 137, pp. 43–90, 2002.

[8] D. Margatitis, "Learning bayesian network model structure from data," *Doctoral Dissertation, School of Computer Science, Carnegie Mellon University*, 2003.

[9] I. Friedman, N.; Nachman and D. Peer, "Learning bayesian networks structure from massive dataset: the sparse candidate algorithm," *Proceedings of the Fifteen Conference on Uncertainty Artificial Intelligence*, pp. 206–215, 1999.

[10] L. E. Tsamardinos, I.; Brown and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Machine Learning*, 2006.

[11] A. R. e. a. E. Segal; D. Pe'er, "Learning module networks," *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 525–534, 2003.

[12] Y. F. Zeng and K. L. Poh, "Block learning bayesian network structures from data," *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*, pp. 14–19, 2004.

[13] Z. Xie, X. C.; Geng and Q. Zhao, "Decomposition of structural learning about directed acyclic graphs," *Artificial Intelligence*, no. 170, pp. 422–439, 2006.

[14] U. e. a. Olesen, K.G.; Kjaerulff, "A munin network for the median nerve - a case study in loops," *Applied Artificial Intelligence*, no. 3, pp. 385–404, 1989.

[15] J. Gross and Y. J, *Handbook of graph theory*. CRC Press, 2003.

[16] M. Koivisto and K. Sood, "Exact bayesian structure discovery in bayesian networks," *Journal of Machine Learning Research*, vol. 5, pp. 549–573, 2004.

[17] D. Margaritis, "Distribution-free learning of bayesian network structure in continuos domains," *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, July 2005.

[18] C. Gonzales and N. Jouve, "Learning bayesian networks structure using markov networks," *Third European Workshop on Probabilistic Graphical Models*, 2006.

[19] Z. Kim and R. Nevatia, "Learning bayesian networks for diverse and varying numbers of evidence sets," *Proceedings of the 17th International Conference on Machine Learning*, pp. 479–486, 2000.

[20] J. e. a. Campos, L; Gamez, "Learning bayesian networks by ant colony optimisation: searching in two different spaces," *Mathware and Soft Computing*, no. 2-3, pp. 251–268, 2002.

[21] L. G. Prashant Doshi and J. C. et. al, "Towards effective structure learning for large bayesian networks," *Proceedings of the Workshop on Probabilistic Approaches in Search, Eighteenth National Conference on Artificial Intelligence*, pp. 16–22, July 2002.

[22] D. H. D.M. Chickering and C. Meek, "A bayesian approach to learning bayesian networks with local structure," *Proc. 13th Conf. on Uncertainty in Artificial Intelligence (UAI'97)*, pp. 80–89, 1997.

[23] T. Silander and P. Myllymaki, "A simple approach for finding the globally optimal bayesian network structure," *22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

[24] K. B. L. J. W. Myers and T. Levitt, "Learning bayesian networks from incomplete data with stochastic search algorithms," *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 476–485, 1989.

[25] K. Murphy, "A brief introduction to graphical models and bayesian networks," *Online article*, 2000.

[26] D.; and H. Lenz, "Learning bayesian networks is np-complete," *Learning from Data: Artificial Intelligence and Statistics*, 1996.

[27] K. Korb and A. Nicholson, *Bayesian Artificial Intelligence*. Chapman and Hall, 2003.

[28] J. G. M. e. a. Abellan, "Some variations on the pc algorithm," *Third European Workshop on Probabilistic Graphical Models*, 2006.

[29] A. Brown and I. Tsamardinos, "A novel algorithm for scalable and accurate bayesian network learning," *Discovery Systems Laboratory, Department of Biomedical Informatics, Vanderbilt University, USA*, 2004.

[30] Wonng and Leung, "An efficient data mining method for learning bayesian networks using an evolutionary algorithm-based hybrid approach," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 2004.

[31] G. Provan and M. Singh, "Learning bayesian networks using feature selection," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1995.

[32] R. C. K. Sivakumar and H. Kargupta, "Learning bayesian network structure from distributed data," *Proceedings of the SIAM International Conference in Data Mining, San Franciso, USA*, pp. 284–288, 2003.

[33] G. Li and T.-Y. Leong, "A framework to learn bayesian networks from changing, multiple-source biomedical data," *Proceedings of the 2005*

AAAI Spring Symposium on Challenges to Decision Support in a Changing World, Stanford University, CA, USA, pp. 66–72, 2005.

[34] D. A. N. e. a. U. Alon, N. Barkai, "Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays," *PNAS*, vol. 96, pp. 6745–6750, June 1999.

[35] P. T. e. a. T. R. Golub, D. K. Slonim, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531–537, October 1999.

[36] M. Carrillo and R. Cantu, "Learning bayesian network structures from small datasets using simulated annealing and bayesian score," *Proceedings of Artificial Intelligence and Applications*, no. 453, 2005.

[37] M. J. D. Agnieszka Onisko and H. Wasyluk, "Learning bayesian network parameters from small data sets: Application of noisy-or gates," *International Journal of Approximate Reasoning*, vol. 2, no. 27, pp. 165–182, 2001.

[38] A. Onisko, "Probabilistic causal models in medicine: Application to diagnosis of liver disorders," *Ph.D. Dissertation, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw*, 2003.

[39] A. Y. W. Wai-Ho Au; Chan, K.C.C.; Wong, "Attribute clustering for grouping, selection, and classification of gene expression data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, pp. 83–101, April-June 2005.

[40] C. T. D. Jiang and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, pp. 1370–1386, November 2004.

[41] S. K. L.J. Hejer and S. Yooseph, "Exploring expression data: Identification and analysis of coexpressed genes," *Genome Research*, vol. 9, pp. 1106–1115, 1999.

[42] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July, October 1948.

[43] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Telecommunications and Signal Processing, Hoboken, New Jersey, USA: Wiley-Interscience, 2nd ed., 2006.

[44] C. J. Date, *Database in Depth: Relational Theory for Practitioners*. O'Reilly Media, Inc., first ed., May 2005.

[45] M. L. X. C. X. L. B. Ma and P. Vitanyi, "The similarity metric," *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 863–872, 2003.

[46] A. K. H. S. R. G. Andrzejak and P. Grassberger, "Hierarchical clustering based on mutual information," *ARXIV Quantitative Biology*, pp. 1–11, December 2003.

[47] S. Watanabe, "Information theoretical analysis of multivariate correlation," *IBM Journal or Research and Development*, vol. 4, no. 1, pp. 66–82, 1960.

[48] M. Studenty and J. Vejnarova, "The multiinformation function as a tool for measuring stochastic dependence," *Learning in Graphical Models*, pp. 261–297, 1998.

[49] W. McGill, "Multivariate information transmission," *Information Theory, IEEE Transactions on*, vol. 4, pp. 93–111, September 1954.

[50] A. Jakulin and I. Bratko, "Analyzing attribute dependencies," *PKDD*, vol. 2838, pp. 229–240, 2003.

[51] A. Jakulin and I. Bratko, "Testing the significance of attribute interactions," *21st International Conference on Machine Learning*, pp. 409–416, 2004.

[52] T. Vilmansen, "On an approximation of mutual information," *IEEE Transactionon Computers*, vol. C-23, no. 12, pp. 1311–1313, 1974.

[53] I. N. N. Friedman and D. Peer, "Learning bayesian network structures from massive datasets: The sparse candidate algorithm," *Proceedings of the Fifteenth Conference on Uncertainty in Articial Intelligence (UAI99)*, pp. 206–215, 1999.

[54] P. Comon, "Independent component analysis, a new concept?," *Signal Processing: Special issue on Higher-Order Statistics*, vol. 36, pp. 287–314, April 1999.

[55] A. M. K. Vilalta R. and E. C. F., "Independent component analysis, a new concept?," *Third IEEE International Conference on Data Mining (ICDM)*, pp. 673–676, November 2003.

[56] A. Hyvärinen, "New approximations of differential entropy for independent component analysis and projection pursuit," *Advances in Neural Information Processing Systems*, vol. 10, pp. 273–279, 1998.

[57] S. W. Y.Y. Yao and C. Butz, "On information-theoretic measures of attribute importance," *Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'99)*, pp. 133–137, April 1999.

[58] R. C. I. Beinlich, G. Suermondt and G. Cooper, "The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks," *Proceedings of the 2nd European Conference on AI and Medicine*, 1989.

[59] C. C. K.; and L. C. N.;, "Approximating discrete probability distributions with dependence trees," *IEEE Transaction on Information Theory*, no. 12, pp. 462–467, 1968.

[60] D. Pelleg and A. Moore, "Dependency trees in sub-linear time and bounded memory," *The VLDB Journal The International Journal on Very Large Data Bases*, vol. 15, no. 3, pp. 250–262, 2006.

[61] R. Sedgewich, *Algorithms in Java Part 5 Graph Algorithms*. Addison Wesley, 1 ed., 2004.

[62] J. Han and K. Kamber, *Data Minning: Concepts and Tecniques*. San Francisco: Academic press, 2001.

[63] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press*, no. 1, pp. 281–297, 1967.

[64] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data" An Introduction to Cluster Analysis*. New York: John Wiley and Sons, 1990.

[65] V. O. Ying Xu and D. Xu, "Minimum spanning trees for gene expression data clustering," *Genome Informatics*, vol. 12, pp. 24–33, 2001.

[66] V. O. Ying Xu and D. Xu, "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees," *Bioinformatics*, vol. 18, no. 4, pp. 536–545, 2002.

[67] Y. Z. O. Grygorash and Z. Jorgensen, "Minimum spanning tree based clustering algorithms," *IEEE International Conference on Tools with Artificial Intelligence*, 2006.

[68] M. K. T. Asano, B. Bhattacharya and F. Yao, "Clustering algorithms based on minimum and maximum spanning trees," *Proceedings of the fourth annual symposium on Computational Geometry*, pp. 252–257, 1988.

[69] B. Ye and K. M. Chao, *Spanning Trees and Optimization Problems*. London: Chapman and Hall, 2004.

[70] C. Zahn, "Graph theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions in Computers*, no. 20, pp. 68–86, 1971.

[71] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256, June 1957.

[72] A. R. and B. A.-L, "Statistical mechanics of complex networks," *Modern Physics*, no. 74, pp. 47–97, 2002.

[73] D. b. A. R. Cohen and S. Havlin, *Structural Properties of Scale Free Networks*. Handbook of graphs and networks, chp. 4, Berlin GmbH: WILEY-VCH, 2002.

[74] T. O. Project, "Online resource," *http://www.opte.org/maps/*, 2007.

[75] N. Paivinen, "Clustering with a minimum spanning tree of scale-free structure," *Pattern Recognition Letters*, no. 26, pp. 921–930, 2005.

[76] R. C. Prim, "Shortest connection networks and some generalisations," *Bell System Technical Journal*, no. 36, pp. 1389–1401, 1957.

[77] J. Gallian, "Dynamic survey of graph labeling," *Elec. J. Combin.*, vol. 14, no. 6, 2007.

[78] R. Neapolitan, *Foundations of Algorithms Using C++ Pseudocode*. Jones and Bartlett Publishers, 3 ed., 2003.

[79] D. Danks, "Learning the causal structure of overlapping variable sets," *Proceedings of the 5th International Conference on Discovery Science*, no. 2534, pp. 178–191, 2002.

[80] Y.-L. Chen and H.-L. Hua, "An overlapping cluster algorithm to provide non-exhaustive clustering," *European Journal of Operational Research*, vol. 173, pp. 762–780, September 2005.

[81] J. Galtier and S. Lanteri, "On overlapping partitions," in *International Conference on Parallel Processing*, pp. 461–468, 2000.

[82] H. Expert, "Hugin," *Online Resource*, 2007.

[83] C. Aliferis, I. Tsamardinos, A. Statnikov, and L. Brown, "Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery," 2003.

[84] K. Kristensen and I. A. Rasmussen, "Preliminary model for barley developed under the project : Production of beer from danish malting barley grown without the use of pesticides," *Online Resource*, 2000.

[85] A. Onisko, "Probabilistic causal models in medicine: Application to diagnosis of liver disorders," *Ph.D. Dissertation, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw, March*, 2003.

[86] e. a. Abramson, B.; J. Brown, "Hailfinder: A bayesian system for forecasting severe weather," *International Journal of Forecasting*, vol. 1, no. 12, pp. 57–72, 1996.

[87] E. J. H. D. E. Heckerman and B. N. Nathwani.;, "Toward normative expert systems: Part i the pathfinder project," *Journal of Methods of Information in Medicine*, pp. 90–105, 2000.

[88] Y. Zheng and C. Kwoh, "Improved mdl score for learning of bayesian networks," *International Conference on Artificial Intelligence in Science and Technology*, 2004.

[89] W. Lam and F. Bacchus, "Learning bayesian belief networks: An approach based on the mdl principle," 1994.

[90] J. Cheng, D. A. Bell, and W. Liu, "Learning belief networks from data: An information theory based approach," in *CIKM*, pp. 325–331, 1997.

[91] G. Elidan, "Bayesian network repository," *Online Resource*, 2007.

[92] Genie and Smile, "Genie and smile bayesian network repository," *Online Resource*, 2007.

Appendix A: Relevant Bayesian Networks

Following some of the Bayesian networks used in this thesis are presented. The datasets presented in this thesis were generating these BNs and the efficient HUGIN [82] system. Some of this networks were obtained from the Bayesian network repository in [91] and in the Genie and Smile website [92].

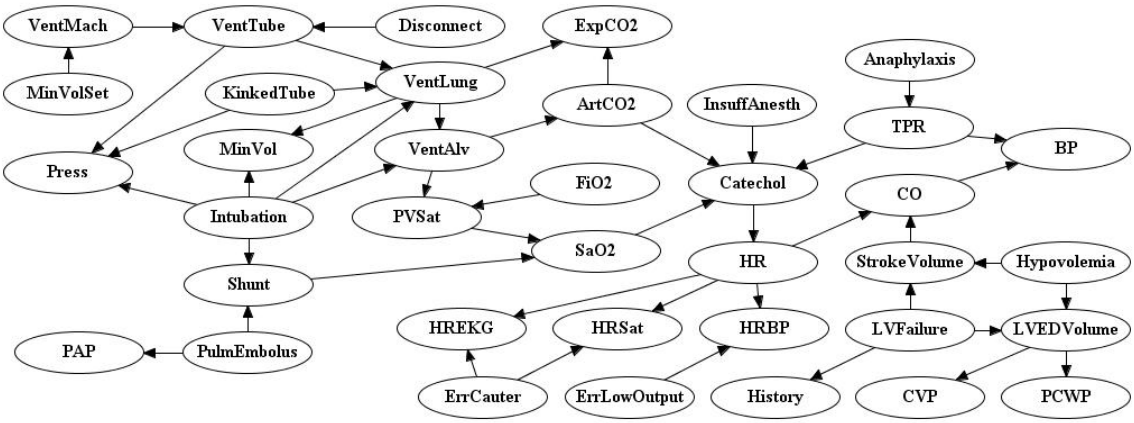


Figure 1: The ALARM Bayesian network.

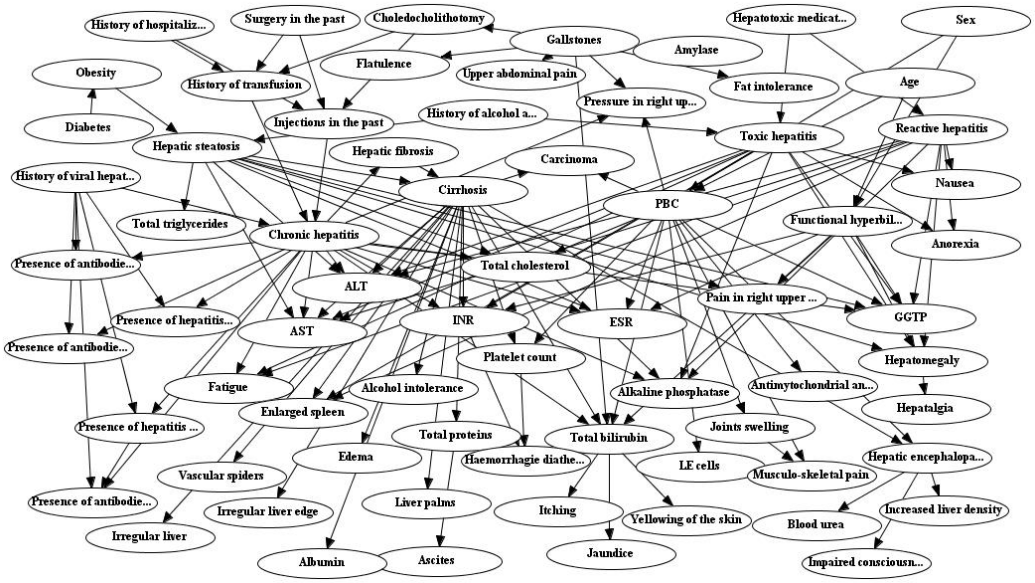


Figure 2: The HEPARII Bayesian network.

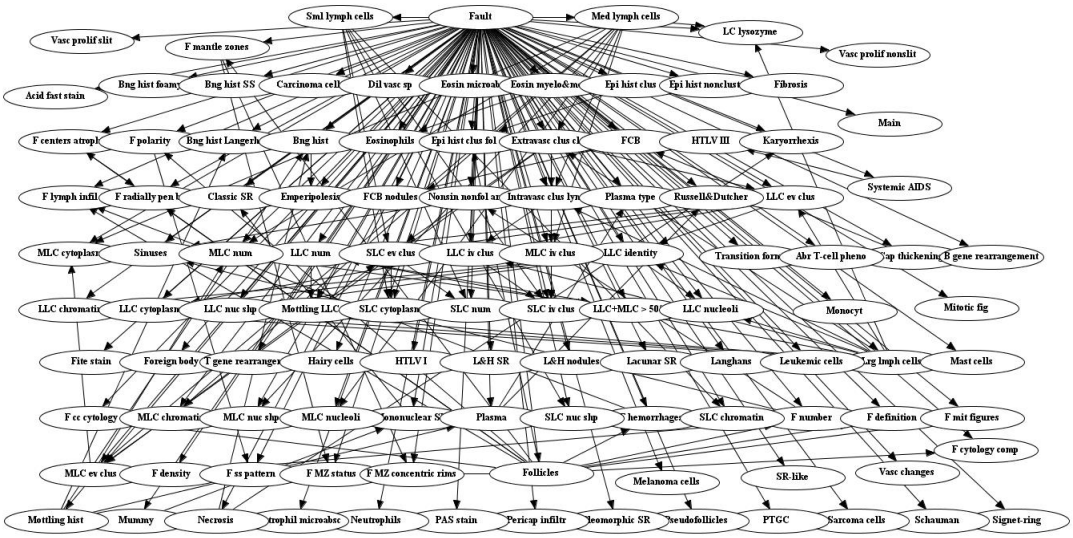


Figure 3: The PATHFINDER Bayesian network.

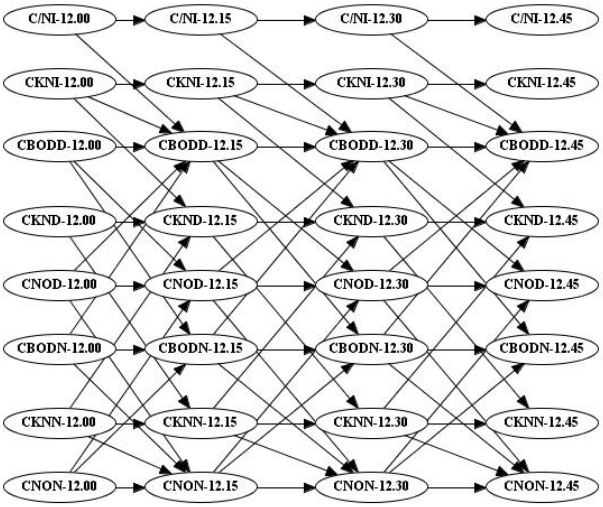


Figure 4: The WATER Bayesian network.

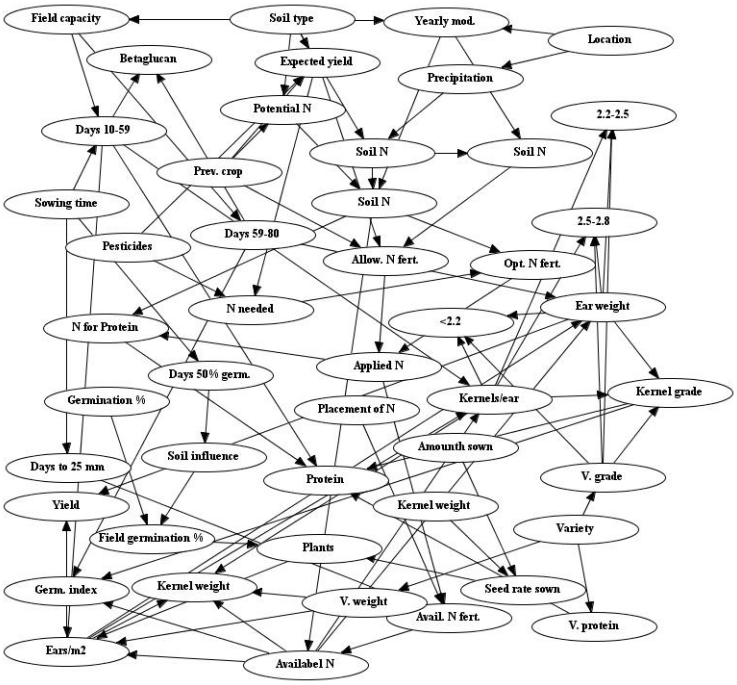


Figure 5: The BARLEY Bayesian network.

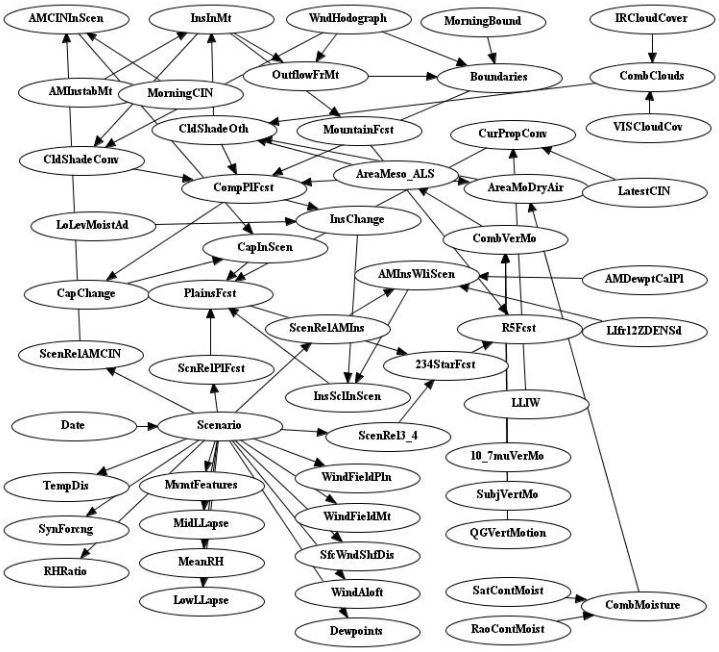


Figure 6: The HAILFINDER Bayesian network.

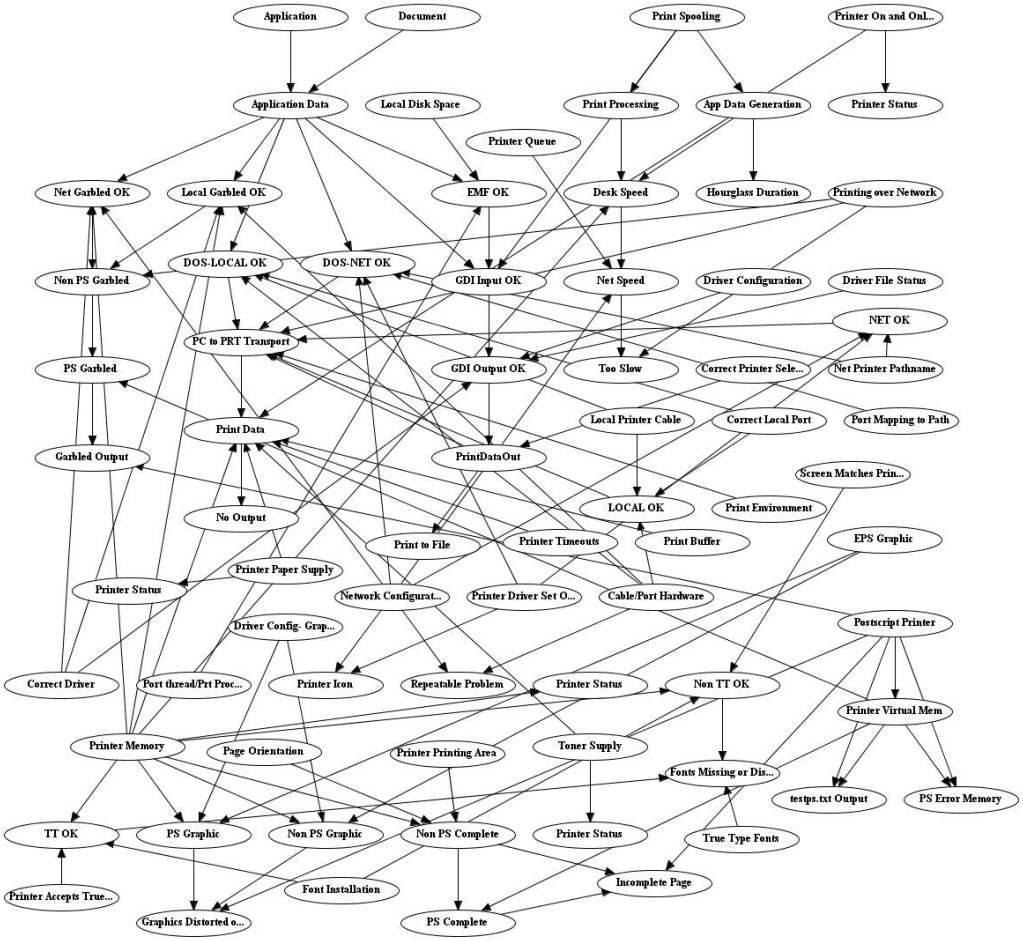


Figure 7: The WIN95PTS Bayesian network.

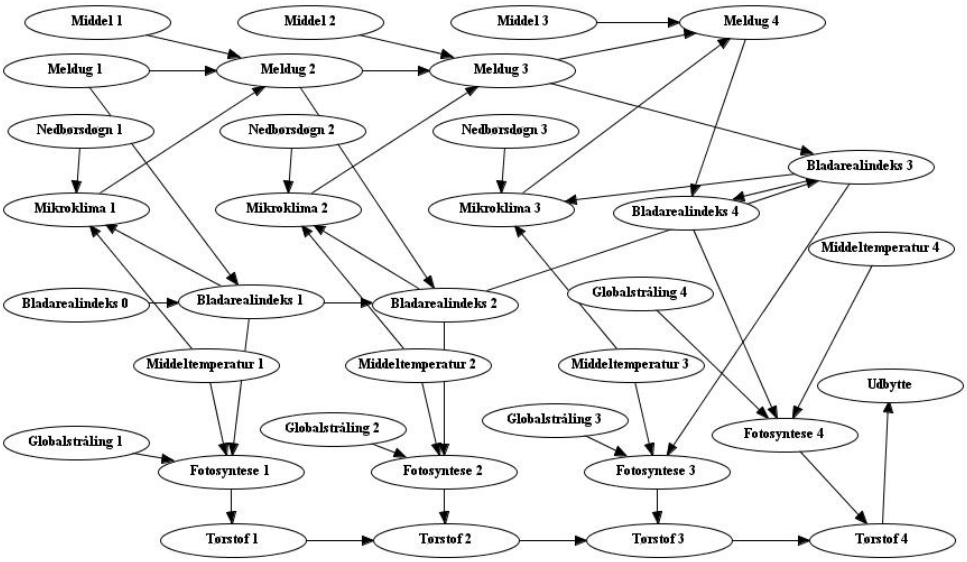


Figure 8: The MILDEW Bayesian network.