Universiteit Leiden
The Netherlands

Opleiding Informatica

Learning Bayesian Networks for

Causal Discovery from Medical Data

Naoufal A. Haddou

Supervisors:
Prof.dr. P.J.F. Lucas & Prof.dr. F.J. Verbeek

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

13/08/2018

Abstract

Causal Bayesian networks are a great tool to capture structure in data, in particular in the medical field. First, they offer a good overview of the probability distribution of the variables. Secondly, they can be used to make predictions for a certain patient or group of patients. Finally, having a causal Bayesian network can help to understand the causal relations between variables better.

However, usually the complete network structure of a problem is not known, whereas without a known network structure one cannot build a Bayesian network from a certain dataset. Fortunately, since the early 1990s several algorithms have been developed that can be used to learn causal Bayesian network structures from a given dataset. These algorithms can roughly be put into two main categories: (1) the *constraint-based*, and (2) the *score-based* algorithms.

In the thesis we compare four of such network learning algorithms, namely: the grow-shrink (GS) algorithm, the Inter Association Markov Blanket (IAMB) algorithm, the hill-climb (HC) algorithm, and the TABU search algorithm.

We were able to establish the performance of all these algorithms, and from that we were able to select the best performing one. We used the best performing algorithm together with medical domain knowledge to build a causal Bayesian network for a medical dataset of patients with a certain type of cancer, called ovarian cancer.

Contents

1 Introduction **1**

2 Related work **3**

3 Preliminaries and background **5**

3.1 Bayesian networks . . . 5

3.1.1 Definition and properties of Bayesian networks . . . 5

3.1.2 Dependence and independence relations . . . 8

3.1.3 Markov blanket . . . 10

3.1.4 Direction of arrows in Bayesian networks . . . 11

3.2 Causal Bayesian networks . . . 14

3.2.1 Interventions . . . 14

3.2.2 Learn causal relations normal vs causal Bayesian networks. . . . 15

3.3 Learning Bayesian networks from data . . . 16

3.3.1 Complete and incomplete data . . . 17

3.3.2 Discrete and continuous data . . . 17

3.3.3 Expert knowledge . . . 18

3.3.4 Algorithms . . . 18

4 Description of the algorithms **21**

4.1 Grow-shrink and incremental association algorithm . . . 21

4.2 Hill-climbing and TABU-search algorithm . . . 23

5 Some experiments with synthetic datasets **26**

5.1 Background of the tests . . . 26

5.2 Description of the algorithm tests . . . 27

5.3 Parameter settings . . . 28

5.3.1 Constraint based algorithm parameters . . . 28

5.3.2 Score based algorithm parameters . . . 29

5.4 Results . . . 29

5.5 Influence of added background knowledge on network score . . . 32

5.6 TABU parameter test . . . 39
5.7 Conclusions . . . 40

6 Description of the medical problem domain **41**
6.1 Symptoms and diagnosis . . . 42
6.2 Treatment . . . 42
6.3 Survival . . . 42
6.4 Motives for building a BN for our medical data . . . 43

7 Learning the network from the gynecology dataset **45**
7.1 The dataset . . . 45
7.2 Network construction using TABU . . . 46
7.3 Clinical use of the ovarian cancer Bayesian network . . . 50
7.3.1 Prior Bayesian network . . . 50
7.3.2 Prognostic use of the Bayesian network on ovarian cancer . . . 51
7.3.3 Subpopulation description . . . 52

8 Conclusion and future work **54**

9 Tables **57**

Bibliography **59**

Chapter 1

Introduction

Discovering relationships between variables from data is at the heart of data science, as in the end what is of primary interest is in what way variables interact and influence each other. A typical method to visualize the interaction between variables is offered by graphs. Variables act as nodes in the graph and the interaction is indicated by means of lines and various types of arrows. The way in which the influence between variables is quantified depends on the formalism that is being used to express this influence. A very popular method in data science to express influence is offered by probability theory. *Probabilistic graphical models* offer a method in which it is visualized by means of graphs how variables interact, whereas the actual influence is expressed in terms of probability theory [7]. Bayesian networks are examples of probabilistic graphical models, where the interaction is shown by means of lines with an arrow-head, so-called directed edges or arcs. Bayesian networks and how they can be learned from data and used for causal discovery is the topic of this thesis.

Bayesian networks are not the only type of probabilistic graphical model; for example, Markov networks, also called Markov random fields, where interactions between variables are represented by means of lines only, i.e. there are no arrow-heads, are also probabilistic graphical models. However, in this thesis, we will be mainly concerned with Bayesian networks.

Bayesian networks have attracted considerable interest in the biomedical field (e.g. [8]). This is not surprising because understanding the way how variables influence each other is one of the main research issues of biomedicine. In particular the interpretation of Bayesian networks as *causal networks* has attracted much attention. This interpretation basically means that a relationship $A \rightarrow B$ is interpreted as saying that '*A causes B*'. However, as will be discussed later in this thesis, it is often not possible to attach such an interpretation to arcs in Bayesian networks.

The purpose of this thesis is to investigate the causal interpretation of Bayesian networks in the context of learning from data. Here there is also a role of prior knowledge, and this is an aspect that is also studied.

We shall consider both simple example datasets that are frequently used by the Bayesian network research community, but later also a real-world clinical dataset will be investigated. Here again we are especially

interested into the question whether causal relationships can be discovered.

This thesis is organized as follows. In Section 2, some related research is briefly summarized. Subsequently, in Section 3 we describe in some detail what Bayesian networks are and how they are learned from data. In Section 4 the functionality of the algorithms we use in this thesis is explained. In section 5 several tests are executed to find the best performing algorithm and its parameter settings. In section 6 the domain of the target dataset, the ovarian carcinoma dataset, is described and relevant information about that domain is given, and then in section 7 best performing algorithm from section 5 is used to build a network for our target dataset. Finally, in section 8 conclusions and remarks for future work are given.

Chapter 2

Related work

Bayesian networks, originally called *belief networks*, were first introduced by Judea Pearl and his PhD students at the beginning of the 1980s [6, 9]. Pearl's seminal book "Probabilistic Reasoning in Intelligent Systems" [10], one of the highest cited publications in artificial intelligence, summarizes most of his research between 1980 and 1988. Basically, the book states that Bayesian networks can be used to represent and reason about generic causal knowledge of a given domain and to draw conclusions given specific input.

After that more was written about the use and properties of Bayesian networks. A more elaborate, technical description is given in [7] and [4].

It was only at the beginning of the 1990s that researchers started to investigate the construction of Bayesian networks fully based on data. Some of the earliest works about creating Bayesian networks from data are by Buntine [2] in 1991, Cooper and Herskovits in 1992 [3] and Heckerman in 1995 [5]. Cooper and Herskovits learning algorithm, called K2, has become quite famous: a network is learned from complete data using a score based method. In the work of Bouckaert in 1994 [1] an other scoring algorithm is used, namely the MDL score. All the works mentioned above assume the given data is complete, i.e. has no missing values.

In the work of Heckerman in 1997 [16] [21] methods are discussed that can work with incomplete data.

Since the first Bayesian network constructing algorithms were described, numerous new algorithms were proposed to create causal Bayesian networks from data.

There are two main approaches these algorithms follow: the score and the constraint based approach. The score based approach can simply be seen as a search algorithm combined with a score function. This score function tells us how good a certain network is, and based on that the algorithm decides how the search will continue. In the constraint based approach independence tests are preformed between the variables in order to find the network structure. The algorithm describes when and on which variables these tests are performed.

There are numerous search algorithms, and from those there are many which work well when used in the score based approach. The same holds for the algorithms that use the constraint based approach, all of them try to

find (in)dependencies between variables in the most efficient way possible. That is why there are numerous of these Bayesian network constructing algorithms.

Most works dig deep into the performance of one certain algorithm. The algorithm can be taken from previous work, or they purpose a new algorithm.

An example of an algorithm that uses the constraint based approach, which we use in this paper, is described in [13] by Dimitris Margaritis. Here a new algorithm, the grow-shrink algorithm, is described and tested.

A score based algorithm, called hill-climbing, is discussed and evaluated in Heckerman 1995 [5].

In our work we test the performance of a selection of the existing algorithms, and we use the algorithm that performs best to create a network for our target data.

There are some works that put existing algorithms to the test, to define which performs the best. One work that compares three algorithms, which are also compared in this paper (namely GS, HC and TABU), is the master thesis of J. D. Leegon [17]. In his thesis six Bayesian network constructing algorithms, from both the constraint based (e.g. GS) and score based (e.g. HC), are compared performance wise in order to find the best performing algorithm.

Two other works that compare algorithms performances are the master thesis of L. D. Fu in 2005 [18] and the research paper of Akwaa and Alkhawlani [19].

In Fu's work certain algorithms that learn networks from continuous data are compared.

In the research paper of Akwaa Alkhawlani [19] five algorithms are compared including three variations of the score based algorithm called K2.

One work that uses algorithms from the Bnlearn package is the PhD work of Marco Scutari in 2011 [20]. Here different algorithms and methods that learn Bayesian network structures from the Bnlearn package are tested. Including the HC, TABU, GS algorithms and the BIC evaluation function.

The main difference between our paper and these above works that compare algorithms performances, is that we execute experiments in search for a good algorithm to use for our target data. The tests we perform are designed to get an insight in this. Whereas in the other works the overall performance of algorithms is compared.

Chapter 3

Preliminaries and background

Knowledge about Bayesian networks that is needed for this thesis will be described in this Section. First, what Bayesian networks are is defined, followed by a description of how their network structure and probability distribution can be learned from data.

3.1 Bayesian networks

To create a Bayesian network with certain variables from a given domain we need information on how the variables interact, i.e., the network structure, and a quantification of this interaction in terms of probabilistic parameters. Both the structure and parameters can be obtained from expert knowledge, or they can be learned from data. Both will be described in this paper.

Next a formal definition of Bayesian networks is given.

3.1.1 Definition and properties of Bayesian networks

A *Bayesian network* $\mathcal{B} = (G, P)$ is defined as a pair, with $G = (V(G), A(G))$ an acyclic directed graph, with $V(G)$ a set of *vertices* or *nodes*, and $A(G) \subseteq V(G) \times V(G)$ a set of *arcs* or *directed edges*. In addition, there is a *joint probability distribution* P associated with the graph that represents to the nodes. The joint probability distribution is defined in correspondence of the graph structure as follows:

$$P(X_{V(G)}) = \prod_{i \in V(G)} P(X_i \mid X_{\pi(i)}) \tag{3.1}$$

where $i \in V(G)$ represents a node in the graph, X_i is the corresponding random variable in the probability distribution P, and $\pi(i)$ are the immediate parents of node i in graph G.

The probability distribution of a variable indicates how likely it is that the variable has one of his values, given that a set of other variables have certain values. The distribution of all variables $X_1, \ldots, X_n$ are written together in the joint probability distribution, as seen above in Equation (3.1).

If node i has no parent nodes the probability is distributed over the values that variable X_i can take. As seen in the joint probability distribution formula, we just get the probability distribution of X_i: $P(X_i)$.

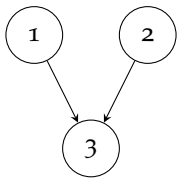


Figure 3.1: Example of a v-structure in a Bayesian network

Example 3.1.1. *Consider the Bayesian network graph shown in Figure 3.1. Its corresponding joint probability distribution is defined as follows:*

$$P(X_1, X_2, X_3) = P(X_3 \mid X_1, X_2)P(X_1)P(X_2)$$

A Bayesian network is build for a finite set of variables X. In the Bayesian network every variable from set X is represented by a node i from the set of nodes $V(G)$. There is a one to one correspondence between nodes from $V(G)$ and the variables from the set of variables X. When we consider a certain node i from network G, the corresponding variable is X_i.

Suppose we have nodes $i, j \in V(G)$. Then i is said to be the *parent* (node) of node j when $(j, i) \in A(G)$, and then node j is the *child* (node) of i.

Node j is said to be a *descendant* of node i if there exists a directed path from j to i. And in that case i is an *ancestor* of j. When there is no directed path from j to i, then j is a non-descendant or i (not an ancestor).

Variables can be categorized in two categories, *continuous* variables and *discrete* variables. A discrete variable can take a value from a (finite) set of countable values. The values of *discrete* variables can be of type: Boolean, ordinal and numeric. Variables with ordinal values have a (ordered) predefined set of values, for example, {cold, warm, hot}. Boolean can be seen as a special type of ordinal with only two values {True, False}. A variable with numeric values is discrete when the variable is limited to a set of countable numeric values.

A *continuous* variable can take a value from an infinite and uncountable number of values. Usually, continuous variables are of numeric type, and we have a certain interval from where the variable can take any value. For example, suppose we have a variable denoting the weight of certain objects, which can be any real number. We know that the number of real numbers is uncountable, and thus this variable is continuous.

In this paper we will only consider discrete variables.

For every variable X_i, there is a set of *(conditional) probability distributions* $P(X_i \mid X_{\pi(i)})$, one for every set of values of $X_{\pi(i)}$, represented as *conditional probability tables* (CPT).

Example 3.1.2. *Consider the network of* example 3.1*, and its probability distribution. Suppose all variables only take Boolean values* $\{T, F\}$*, and that the three nodes in the network of Figure 3.1, node* 1, 2 *and* 3*, are respectively representing the variables,* X_1*,* X_2 *and* X_3*. Then, the* CPT *of variable* X_3 *may look like the one shown in table* 3.1 *below.*

X_3	$P(X_3 \mid X_1 = T, X_2 = T)$
T	0.4
F	0.6

X_3	$P(X_3 \mid X_1 = T, X_2 = F)$
T	0.1
F	0.9

X_3	$P(X_3 \mid X_1 = F, X_2 = T)$
T	0.2
F	0.8

X_3	$P(X_3 \mid X_1 = F, X_2 = F)$
T	0.3
F	0.7

Table 3.1: Conditional probability distribution in CPT

When node i has one or more parent nodes, the probability is distributed over the values of X_i, for every combination of the values the parent nodes of X_i can take. For every of such distribution the sum over all probabilities must be equal to 1, this can be written as the following equation:

$$\sum_{x_i} P(X_i = x_i \mid X_{\pi(i)}) = 1.$$

When node i has $n \geq 0$ parents, with X_i and its parents $X_{\pi(i)}$ Boolean, there are $2 \cdot 2^n = 2^{n+1}$ elements in the CPT.

We discussed that the probability is distributed based on the network structure. But the actual numbers we fill in for that probability distribution, can either be *known from expert knowledge*, or it can be *learned from data*.

When we have the joint probability of a net, it is possible to calculate all probabilities from joint probabilities using just two rules, using the rule of *marginalization* and the rule of *Bayes*.

With marginalization we sum out variables that we don't want. We can write this marginalization with the following equation:

$$P(X) = \sum_{Y} P(X, Y)$$

Using this rule of marginalization we can sum out every probability from the joint probability distribution, to get the probability we are interested in. Together with the *Bayes* rule we can calculate all probabilities from a given (joint) probability distribution.

The Bayes' rule is written as the following equation:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Using Bayes' rule it is possible to calculate $P(A|B)$ knowing $P(B|A)$ and $P(A)$.

3.1.2 Dependence and independence relations

Independence relations are defined for both the *graphical structure* of the Bayesian network, as for the *probability distribution*. Below we will first describe the independence relation in probability distribution, written as $\perp\!\!\!\perp_P$, where the P denotes that we are considering the probability distribution. Secondly, the independence relation in the graphical structure is described, written as $\perp\!\!\!\perp_G$, where the G denotes that we consider the independence relation of the graph.

Probability independence definition Two sets of variables X and Y can either be *dependent* or *independent* on each other given a third set of variables Z. With Z is given is meant that variable Z is set to one of his values, and we keep this value the same.

When X and Y are independent given Z it means that, when Z is given, and we take the probability distribution of X, then changing the distribution of Y can not result in a change in the distribution of X. So this means that when Z is given, changing the distribution of Y can not influence the probability of X.

This can be written as the following equation:

$$X \perp\!\!\!\perp_P Y \mid Z \text{ is true iff } P(X \mid Y, Z) = P(X \mid Z) \text{ for all values of } X, Y \text{ and } Z$$

Here, sign $\perp\!\!\!\perp_P$ is used to denote an independence relation.

When X and Y are independent given Z, then the probability distribution of X given Z is the same as the probability distribution of X given Y and Z. This should hold for all values of X, Y and Z.

So two sets of variables given a third set must be either independent or dependent. When they are not independent, it means that they are dependent, and when they are not dependent, it means that they are independent.

Thats why, in Bayesian networks, the set of *independence relations* is used to denote the dependencies between the variables. This give us enough information about how the variables depend on each other.

From the definition of independence, thus follows the definition of dependence, which is the exact opposite of probability independence.

When X and Y are *dependent* given Z it means that, when Z is given, and we take the probability distribution of X, then changing the distribution of Y can result in a change in the distribution of X. So the distribution of Y has an influence on how the probability of X is distributed. We can summarize this with the following equation:

$$X \not\perp\!\!\!\perp_P Y \mid Z \text{ is true iff } \exists(X, Y, Z) \text{ where } P(X \mid Y, Z) \neq P(X \mid Z)$$

Graphical independence definition Here, the independence relation between sets of nodes is defined based on the graphical structure.

Two sets of nodes I and J, where a third set of nodes K contains nodes which represent variables that are given, are said to be *independent* given K when set K *d-separates* nodes I from nodes J.

A set K is said to *d-separate* I from J if and only if K blocks every path from a node in I to a node in J. We can write this as the following equation:

$$I \perp\!\!\!\perp_G J \mid K \text{ is true iff } K \text{ d-separates all paths } p \text{ from } I \text{ to } J$$

A path p is said to be d-separated (or blocked) by a set of nodes K if and only if:

1. *p contains one of the structures $i \to m \to j$ (called a chain) or $i \leftarrow m \to j$ (called a fork), then middle node m must be in K, or*
2. *p contains structure $i \to m \leftarrow j$ (inverted fork), then the middle node m must not be in K, and by this no descendant of m is in K.*

In the same way as for the independence between the probability distribution of variables, two sets of nodes can either be dependent or independent of each other given a third set. When they are not independent they must be dependent, and the same hold the other way around.

From this independence relations set we can deduce the dependence relations.

When two sets are dependent the following holds:

$$I \not\perp\!\!\!\perp_G J \mid K \text{ is true iff } K \text{ does not d-separate all paths } p \text{ from } I \text{ to } J$$

When two sets are dependent given a third set, we thus can conclude that not all paths are blocked by set K.

Mapping between probability and graphical independence relations We have defined independence between variables based on their probability, and independence between nodes based on the structure of the graph. When certain in-dependencies are known, either in the probability $\perp\!\!\!\perp_P$ or in the graphical structure $\perp\!\!\!\perp_G$, they can be translated to the other form.

In order to translate $\perp\!\!\!\perp_P$ we simply have to use the nodes represented by the variables in $\perp\!\!\!\perp_P$, to get our $\perp\!\!\!\perp_G$. For example, X_i becomes i.

It may be the case that a $\perp\!\!\!\perp_P$ is valid, by which we mean that it is true that there is an independence relation between the sets of variables in $\perp\!\!\!\perp_P$. But the independence does not hold when translated to the graphical form $\perp\!\!\!\perp_G$. Or that we have a certain $\perp\!\!\!\perp_G$, and when translated to $\perp\!\!\!\perp_P$, we do not get the same independence as for $\perp\!\!\!\perp_G$.

Thats why there are three different types of mappings defined between the probability distribution and the graphical structure.

Suppose P is the probability distribution for a certain set of variables X, and graph $G = (V(G), E(G)$ is an directed graph created for set X, then for each mutually disjoint set $U, V, W \subseteq V(G)$:

- G is called an *directed dependence map*, in short *D-map*, if:

$$X_U \perp\!\!\!\perp_P X_V | X_W \Rightarrow U \perp\!\!\!\perp_G V | W$$

- G is called an *directed independence map*, in short *I-map*, if:

$$U \perp\!\!\!\perp_G V | W \Rightarrow X_U \perp\!\!\!\perp_P X_V | X_W$$

- G is called an *directed perfect map*, in short *P-map*, if:

$$X_U \perp\!\!\!\perp_P X_V | X_W \Longleftrightarrow U \perp\!\!\!\perp_G V | W$$

So in a *D-map*, all independence relations based on the probability distribution P, are also independence relations in the graphical structure. In an *I-map* this holds in exactly the opposite direction. If G is both a D-map and I-map it is called a *P-map*.

3.1.3 Markov blanket

The *Markov condition* (sometimes called Markov assumption) for a Bayesian network states that: any node in a Bayesian network is conditionally independent of its non-descendent's given its parents [11]. This can be seen as the parents blocking (d-separating) the information from the non-descendent's.

The *Markov Blanket* of a node a, $BL(a)$, is the set of nodes composed of:

- the parents of a,
- the Children if a, and
- the parents of the children of a.

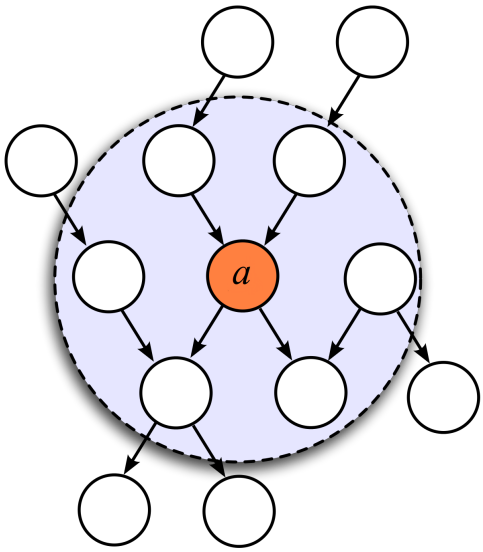


Figure 3.2: The nodes inside the interrupted circle form the *Markov blanket* of node *a*

A node is conditionally independent of the entire network, given its Markov blanket.

The Markov blanket of a node contains all the nodes that shield the node from the rest of the network. This means that the Markov blanket of a node is the only knowledge needed to predict the behavior of that node.

For the Markov blanket of a certain node $n \in W$, the following statement holds:

For any node $n \in W$ with the Markov blanket $BL(n) \subseteq W$ is the minimal set of nodes such that for any $m \in (W \setminus BL(n)) \setminus \{n\}$ we get that $\{n\} \perp\!\!\!\perp \{m\} | BL(n)$, where '$\setminus$' is set-difference.

Knowing the Markov blanket of a certain variable x thus results in the variable being independent all the other variables outside the Markov blanket.

3.1.4 Direction of arrows in Bayesian networks

In probability theory and statistics, *Bayes' theorem* is used to describe the probability of an event based on the (known) probability of other events.

Bayes' theorem is stated mathematically as the following equation:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Bayes theorem is based on the following equation that holds for all Bayesian networks:

$$P(B \mid A) \cdot P(A) = P(A \mid B) \cdot P(B)$$

Suppose a simple network of two nodes A and B, with a single directed edge from A to B. Then the joint

probability distribution is:

$$P(A, B) = P(A \mid B) \cdot P(B)$$

Now suppose we take the exact same network, only now we turn the single edge around. So it now goes from node B to node A. For this network, the joint probability distribution is:

$$P(A, B) = P(B \mid A) \cdot P(A)$$

According to Bayes theorem, these exact probability distributions are equal, and by that we can conclude that both these networks are *equivalent*.

Different Bayesian network structures are said to be *equivalent* when:

- They encode precisely the same conditional independence relations
- They have the same joint probability distributions

Of course, most networks have a more complex structures where the network may consist of large amount of variables and edges. In complex networks, turning the direction of an edge around may result in a change in the independence relations between the variables, and thus a non-equivalent network. But there may also be variables where changing the direction results in an equivalent network.

Whether or not we can change the direction of a particular edge while still keeping an equivalent network is based on how the variables around the edge are structured.

When an edge is part of, what we call, a *v-structure* we can not change the direction of that edge. The v-structure contains causal relations, one variable is the cause and one is the effect.

When a Bayesian network contains a *v-structure*, changing the direction of an edge inside that v-structure results in:

- A change of probability distribution P of the network, and thus a non-equivalent network
- It also results in a change in the independence relations $\perp\!\!\!\perp_G$ between the nodes

Nodes $1, 2, 3$ are said to be structured in a *v-structure* when they are structured as in figure 3.3 below:

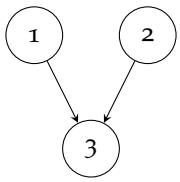


Figure 3.3: Example of a v-structure in a Bayesian network

As we can see in the figure, one of the three nodes must have two incoming directed edges coming from the two other nodes. Looking at he above structure we can state that the following graphical independence

relation holds:

$$\{1\} \not\perp\!\!\!\perp_G \{2\} \mid \{3\}$$

So in this graph node 1 and 2 are dependent given node 3.

When we change the direction of edge $2 \rightarrow 3$ we get the graph structure shown in figure 3.4 below:

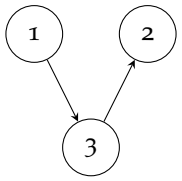


Figure 3.4: Non equivalent network to the one in figure 3

When we rewrite the independence relation between node 1 and node 2, we get the following:

$$\{1\} \perp\!\!\!\perp_G \{2\} \mid \{3\}$$

Now node 1 and 2 are independent given node 3.

So changing the edge inside the v-structure resulted in a change in the independence relations, and thus the network structure resulting from the alteration of the edge is a non-equivalent network.

There is an simple example below to get a better idea about the reason why v-structure can be given a causal interpretation.

Example 3.1.3. *Suppose we have two variables, Rain and WetFloor, which both can take Boolean values True, False. Where variable Rain denotes whether it is raining or not, and WetFloor denotes whether the floor on our lawn is wet or not.*

Suppose the only information we have is that these two variables are correlated to each other. From this information alone we can not decide whether Rain causes a WetFloor or that WetFloor causes Rain.

Now the idea with the v-structure means the following. When we observe a variable correlated to one of our variables but not correlated to the other, we can create a causal structure from them.

Suppose we observe a third variable, SprinklerOn, which is a Boolean variable denoting whether the sprinkler is on or not.

When we then observe that SprinklerOn is correlated to WetFloor, but not correlated to Rain, we can from this conclude that SprinklerOn and Rain are both causes of WetFloor.

Because SprinklerOn and Rain are not correlated but both correlated to WetFloor, they both cannot be consequences of WetFloor, because then they would be correlated to each other. So this is the only way we can structure them.

These v-structure are a great opportunity, because they can be used to describe real causal relationships in normal Bayesian networks. The algorithms that we use make use of these v-structure to create a normal Bayesian network containing causal relations.

But as we have seen, the direction of the arrow in a normal Bayesian Network does not necessary imply causality. While in some cases, it may be useful to have such a data-structure.

There is a special type of Bayesian network, called a *causal Bayesian network*, where by definition all edges contained in the network are causal.

Next we will discuss how such a *causal Bayesian network* is defined.

3.2 Causal Bayesian networks

Causal Bayesian networks are defined by Judea pearl [11] in order to have a type of Bayesian network data-structure that encodes solely causal relations. Normal Bayesian networks represent probabilistic independence, whereas *causal Bayesian networks* represent causal relationships.

3.2.1 Interventions

The notion of *intervention* is used in the definition of causal Bayesian networks. Intervention simply means that we assume that we perform the action that results in one of the variables getting a certain value.

When intervention is done on a certain variable X_U (where U is the node representing the variable in a certain graph G), all edges from the parents of U to U are deleted, and X_U is set to one of his values. For example, if X_U is a Boolean variable with values {true,false} then we can set $X_U = true$. The reason why these edges are removed is because it is certain that we are the sole cause that resulted in the node getting the value.

When intervention is done on X_U it is possible that the probability distribution of the variables around the intervened node G changes.

When a DAG G build for probability distribution $P(X)$ is said to be *causal Bayesian network* it means that for every *intervention* $T \in X$ that is possible to do on one of the set of variables X, the probability distribution of certain variables must not be affected by the intervention (it must remain unchanged before and after).

Below a simple definition is given of this intervention property that must hold between variables in a causal Bayesian network [11]:

1. Suppose our prior probability distribution is $P(X)$ of our set of variables X
2. Now for every possible intervention set T that we can possibly create from variables X: $T \subset X$, the following must hold:

(a) First compute the new probability distribution P_t that we get when intervention on variables in intervention set T is done.

(b) Now every node i from set of nodes I from the graph who has at least one parent node contained in T, we take the variable it represents X_i:

(c) Then it must hold that:

$$P(X_i|pa_i) = P_t(X_i|pa_i)$$

The probability distribution of one of these particular variables given its parents after intervention, must be the same as the probability distribution given its parents before intervention

We can see that a causal Bayesian network differs from a normal Bayesian network. As we have seen in a normal Bayesian network it is possible turn the direction of an edge around while still keeping the same probability distribution. In a causal Bayesian network, this is not possible.

We are looking at a causal relationship, one is the cause and one the effect, and we cannot turn around the cause and effects. That becomes more clear with the following example:
Example 3.2.1. *Suppose we have two variables, and a* switch *to turn the light on which can be in state* on / off*, it can be pressed to put it in state on, and pressed again to put it in state off. And with that switch we can turn on the* light *which can be* on,off*. From simple reasoning we can say that the switch may cause the light to be on, or off. Because when we suppose both are functioning, pressing the switch should result in the light going on or off. We thus have a Bayesian network that looks like $S \rightarrow L$. Suppose we now turn the cause and effect around. Then we say the light causes the switch to be on or off, and that does not make any sense. When in some way we can make the light go on without touching the switch, that can not cause the switch to jump from the off to on state.*

3.2.2 Learn causal relations normal vs causal Bayesian networks.

The problem with these causal Bayesian network is that, in order to learn them, *experiments* must be performed in order to test whether relations are causal or not. In an experiment it is actually tested whether one variable is the cause of an other variable, in example 3.2.2 an example of such an experiment is described.

When a network must be learned from data, it is not possible to execute these experiments, and thats why these causal Bayesian networks can not be used to learn causal networks from data.

That is why, in the field of learning causal relations from data, *normal Bayesian networks* are used instead of causal Bayesian networks.

In normal Bayesian networks, v-structures are used to describe causal relations. In order to learn causal relations using a normal Bayesian network the network should be build in such a way that it contains v-structures. As described in the previous section, the structure of a v-structure causes that we cant turn around an edge in such a structure without changing the independence relations of the variables. After we have build

our network, it is possible that we have relations in our network that are not inside a v-structure and thus are not causal. In order to make these relations causal, we use our *expert knowledge*.

By this we mean that we know a causal relation in a certain domain via *expert knowledge*, or other reasoning. In this case we can assume that some variable causes the other. For example, in the example of the room below, we already know that the Thermostat is the cause of the Roomtemp.

Example 3.2.2. *(Thermostat and room temperature) In this experiment suppose that we have a room with a thermostat to heat up the room, and a thermometer to measure the room temperature. We consider only two variables: Thermostat, which can be on or off on,off, and room temperature (Roomtemp), which indicates whether the room temperature is either smaller than* 20 *degrees, or* 20 *degrees or higher. When the thermostat is turned on it keeps on heating up the room, when it is off it does nothing, and we assume that the thermostat is a simple switch with an on and off state, which has to be operated manually.*

We start testing whether room temperature (RoomTemp) is the cause of Thermostat. We can test this by setting Thermostat to a fixed value, and check whether an alteration of Roomtemp can result in a change in this fixed value of Thermostat. When conducting this we thus find that Roomtemp is not the cause of Thermostat. From simple reasoning we know that the change of temperature does not change the state of the switch. The next step is then to test if Thermostat is the cause of Roomtemp. This time we fix the state of Roomtemp, and alter the value of Thermostat. From simple reasoning, the Thermostat can heat up the room, and thus cause the temperature in the room to change, and thus it can cause the value of Roomtemp to change. When conducting this experiment we thus find that Roomtemp is in fact the cause of Thermostat.

3.3 Learning Bayesian networks from data

There are algorithms that learn Bayesian networks from a given dataset.

For the algorithm to be used to build a network for a certain dataset, the data must be represented in a certain way, that the algorithm can use it.

Usually the dataset contains a set of variables X, which will be used by the algorithm to learn the dependencies, and build a graph for. In order to learn dependencies between variables, the dataset must contain a certain amount of records. A *record* is a combination of values the variables had taken. The amount of records a dataset contains plays a role in how good the network the algorithm learn will be, this is further discussed and shown in our tests.

Other things that play a role in Bayesian network generation from data are whether the data has missing values and what type of variables the data contains. This is described below.

Below is a simple example of a dataset.

Example 3.3.1.

Date	*Weather*	*Playing Football*
03/01/2018	*sunny*	*yes*
05/01/2018	*misty*	*yes*
10/01/2018	*sunny*	*no*
12/01/2018	*rain*	*yes*
13/01/2018	*NA*	*no*
15/01/2018	*rain*	*yes*

where 'NA' stands for 'non-available'.

3.3.1 Complete and incomplete data

A dataset can either be complete or incomplete. A dataset is said to be complete if no record contains a missing (N/A) value. Otherwise the data is said to be incomplete.

The dataset in example 3.3.1 above is *incomplete*, because it contains one record with a missing value.

Most datasets used in data analysis are incomplete. Usually when data is gathered, there are records where we can fill in all variables, and there are records where certain variables are not filled in due to certain circumstances. For example, when we consider a dataset created by a store for their clients, we may have the age of certain clients, and not of others.

The missing values may prevent an algorithm from using the data.

There are two ways to deal with missing data. The missing values can either be filled in, in a process called imputation. In imputation the missing values are predicted, based on the rest of the data.

The other way is simply dropping all records that contain missing values. This method must be taken with care, because in this method useful information can be lost. When a large proportion of the records contain missing values, it may not be a great idea to drop all these records, because in this case it results in a loss of a large amount of data. On the other side, when the amount of records containing missing values is negligible, this method may be useful.

There are algorithms that can work with incomplete datasets. These algorithms simply use imputation before using the data.

3.3.2 Discrete and continuous data

Data can contain discrete and continuous variables. For discrete variables, the probability distribution represented by a CPT is build to represent the variable in the network (as described in the above sections). For continuous variables, this cant be done because there is not necessarily a finite amount of values for the variable.

There are usually two methods to use continuous variables in Bayesian networks. The first one is to simply discretize the continuous variables. For example, when we have a continuous variable weight, representing the weight of a planet, then instead of using the actual numeric weight, we can discretize this variable to a discrete variable with two values, larger then 100 million kg and smaller the 100 million kg.

It is not necessary to discretize these variables in order to use them to build a Bayesian network. There are special types of Bayesian networks defined, called Gaussian Bayesian networks, which support the use of continuous variables. In these networks the probability distribution is continuous and based on the linear Gaussian model. A good description of these models is given in "An Introduction to Gaussian Bayesian Networks" [22]. We will not consider this in this paper, so it is not necessary to go further into this

3.3.3 Expert knowledge

From *expert knowledge*, we can (partly) know the structure of a network. From this expert knowledge we can deduce relations between the variables in our domain. These relations can be useful to create a network from our variables.

Algorithms can use this expert knowledge to build a network from data. To do that the expert knowledge has to be represented in the way the algorithm can use it as an input. In the algorithms that we use in this paper, the edges between variables can be inserted into the algorithm using a parameter (more in section 5).

For example, we can know that eating a pancake has no causal influence on what season it is. In this case we have to input into the algorithm that arrow from eating a pancake to season can not be in the network the algorithm creates.

3.3.4 Algorithms

There are several algorithms that support building networks from (in)complete data. There are roughly two categories of Bayesian network building algorithms, the *constraint based algorithms* and the *score based algorithms*. In our tests in section 5 we use two algorithms of each type.

The functionality of these two categories of algorithms is described below.

Constraint based algorithms

The constraint based algorithms are all derivatives of the Inductive Causation algorithm [12]. The core idea of constraint based algorithms is to learn conditional dependencies between the variables from the dataset.

These algorithms first try to find the Markov blankets of all the variables, and then they construct a network where these found dependencies are maintained.

Basically, the working of a constraint based algorithm can be divided in three parts:

- First the independence relations have to be learned from data.
- Then the direction of the edges is determined
- Finally a graph is constructed, that may be partially directed. At the end the direction of the undirected edges has to be done manually.

Score based algorithms

The core idea of a *score based algorithm* is that we use a certain search algorithm to search for the optimal graph.

Simply put, a score-based algorithm uses a search algorithm together with a evaluation function to search for the optimal graph structure. The search algorithm searches for candidate solutions, and the evaluation function decides how good the solution is, and based on that the search function decided how to continue the search.

There are different evaluation functions and they will be described below in this section.

Basically the following happens in these type of algorithms:

1. First we have to decide which graph we start with, empty graph or something else, and we give that graph a score using a chosen evaluation function
2. Then we expand the graph by adding/removing an edge, or by reversing an edge in the graph and we compute the new score
3. When the new score is higher, it means our new graph is better, and we repeat step 2
4. When the new score is smaller, we continue with our old graph and try to add remove or reverse an other edge

The *evaluation function* takes the graph G and dataset $D = \{a_1, \ldots, a_n\}$ as an input and computes the value $S(G, D)$ which tells us how well it fits the data. It can for most scoring functions be seen as roughly a *likelihood function* combined with functions that cover other aspects of evaluating the network.

The *log-likelihood* is an example of such an evaluation function, and it tells us how likely a model is for a certain dataset. The idea is that we create a graph G, and with the dataset we estimate the parameters of G. Then we create a new dataset on the same variables, and we estimate how good our network performs on the new dataset.

The log-likelihood evaluates one aspect of building a good network for a dataset, the aspect of it *fitting* with the dataset.

An other aspect that score functions take into account is the complexity of a network. A less complex network has in most cases the preference over a more complex one.

A score method also gives certain preference to one of the aspects over the other. It may be necessary that the created network must fit the data as good as possible, while the amount of complexity isn't really important till a certain degree.

Below is a evaluation function called the *Bayesian Information Criterion*, shortly BIC, which we will use in this paper. BIC uses the log-likelihood combined with a part that estimates the complexity of a network to compute the score.

$$s(D, N) = -2 \log P(D \mid M) + d \cdot \log N$$

where $N = |D|$, the sample size of the dataset D, d the number of parameters, and M the model used to compute the predictions. The lower BIC score signals a better model. Here the $d * log(N)$ part estimates the complexity.

Chapter 4

Description of the algorithms

We selected four algorithms, which we will compare in section 5, in order to find out which of the four performs best.

From the algorithms perspective, our target dataset is complete, and contains only discrete variables. All of the following algorithms work with datasets with these properties.

Because of time as a limiting factor we decided to compare four algorithms, two of each category.

We decided to have a small selection of algorithms, so we can put more attention in understanding the functionality of every algorithm. Furthermore, for future work, it can be beneficial to know the functionality of the better performing algorithm. Because when this is known better, we can search for better algorithms that are, functionality wise, close to the better algorithm.

Below each algorithm is described.

4.1 Grow-shrink and incremental association algorithm

In the tests we use two constraint based algorithms, namely the *Grow-Shrink (GS)* algorithm and the *Incremental Association Markov Blanket* algorithm. Below the working of both algorithms is explained. For a more elaborate explanation of both algorithms we refer to their documentation and source code.

Grow-Shrink Algorithm In this algorithm the Markov blanket is computed for every variable $x \in X$, using the following steps for every variable [13]:

1. $S \leftarrow \emptyset$
2. While $\exists Y \in X - \{x\}$ such that $Y \not\perp\!\!\!\perp x|S$, do $S \leftarrow S \cup Y$. [Growing phase]
3. While $\exists Y \in S$ such that $Y \perp\!\!\!\perp x|S - \{Y\}$, do $S \leftarrow S - \{Y\}$. [Shrinking phase]

4. $B(X) \leftarrow S$

Here S is used as the set containing the (current) Markov blanket of variable x. Initially, in step 1, this set S is empty.

Then in step 2 S is filled with variables from X. This happens if there exists a variable Y that results in Y and x being dependent given the Markov blanket S. If that is the case then Y is added to the Markov blanket S. This is called the *growing phase* because the Markov blanket is filled, and thus "grows", in this phase.

This happens until there are no more variables that can be added to the Markov blanket.

It is possible that there are too much variables added to the Markov blanket. Thats why in step 3 it is checked whether a variable Y is inside the current Markov blanket, while he is in reality not part of the Markov blanket. That variable is then removed.

And this continues until no more variable can be removed. This is called the *shrinking phase* because variables that are not part of the Markov blanket are removed, and it thus shrinks to the actual Markov blanket.

This algorithm contains the essence of a constraint based algorithm, it used conditional independence tests in order to find the Markov blankets of all variables.

Incremental Association Algorithm Based on the Incremental Association Markov blanket (IAMB) algorithm. This algorithm recovers the Markov blankets of the variables in a way that is similar to the GS algorithm.

The IAMB algorithm consists, like the GS algorithm, of two phases. In the forward phase the Markov blanket is filled with variables that are (possibly) part of the Markov blanket, and in the second phase variables in the Markov blanket are tested on whether they actually are part of the Markov blanket.

Below is the pseudo code of the iamb algorithm.

1: **Phase I** (forward)
2: $CMB = 0$
3: **while** CMB has changed **do**
4: Find the feature Y in $X - CMB - \{x\}$ that maximizes $f(Y \perp\!\!\!\perp x|CMB)$
5: **if** $Y \not\perp\!\!\!\perp x|CMB$ **then**
6: Add Y to CMB
7: **end if**
8: **end while**
9:
10: **Phase II** (backwards)
11: Remove from CMB all variables Y, for which $Y \perp\!\!\!\perp x|CMB - \{Y\}$
12: Return CMB

The Markov blanket has to be computed for every variable $x \in X$, set X contains all variables of the domain. The CMB is a subset of variables form X ($S \subseteq X$), used as the Markov blanket. Initially this set is empty. Then this set is filled in phase 1 (line 3 to 8). Here the heuristic function f is used to select which variables should be put in the Markov Blanket. So from the set containing variables $X - CMB - \{x\}$, we have to find the selection of variables Y from that set, which maximize this heuristic function f.

This function f gives us a measurement of association between Y and x given CMB. In this way, more associated variables are put into the Markov blanket first, and by that, it prevents us to put more variables into the Markov blanket then necessary

When such a set Y is found, it is checked whether Y and $\{x\}$ are dependent given our current Markov blanket CMB. If that is the case, then Y should be added to the Markov blanket. This continues until the Markov Blanket is filled.

Then in the backward phase variables that are in CMB but are not part of the Markov blanket are detected and removed. That is done by checking whether x is independent of one of the variables of the current CMB (here called Y), given the CMB that results when removing the particular variable from the CMB set.

4.2 Hill-climbing and TABU-search algorithm

We will use two score based algorithms. For both of these algorithms we use the same evaluation function. The two algorithms are the *Hill climbing (HC)* algorithm and the *Tabu Search (TABU)* algorithm, and the evaluation functions we use is *The Bayesian Information Criterion (BIC)*. Below the working of both algorithms is roughly explained. For precise explanation of both algorithms we refer to their documentation.

Hill climbing algorithm The *hill climb* algorithm [14] is a *greedy* search algorithm. It starts with a certain initial graph configuration, and from there it creates new configurations by adding, removing or reversing one edge. The algorithm always chooses the new configuration with the highest score. This is repeated until a certain ending condition is met.

This algorithm is defined with the following pseudo-code:

1: Let s be the initial state
2: **while** !$MaxIterations$ **do**
3: Expand current state s by search entire neighborhood by making all possible local change
4: S = X(s)
5: Evaluate each transition state from current neighborhood
6: $\forall t_i \in X(s), \exists i \rightarrow max(e(t_i))$
7: **if** $e(t_i) \geq e(s)$ **then**
8: s=t
9: **else**

10: break
11: **end if**
12: **end while**
13: Return s

This algorithm we starts with a certain graph: The initial state.

Then, the set S is created, which contains all possible configurations that we can create from the initial state by doing one local change (line 4). Such a local change can be either adding, reversing or deleting one edge inside the initial state.

For every configuration it computes the score using the selected evaluation function, and it selects he configuration with the highest score (line 5).

If the highest score found is higher then the score of the initial state, the next configuration is set to this highest found configuration.

These steps are then repeated for a certain amount of iterations.

Tabu search algorithm The *TABU-search* algorithm [15] uses the greedy approach, like the hill-climb algorithm. This algorithm starts with an initial configuration, and from there new configurations are constructed by adding, removing and reversing edges. In the process the score of these are evaluated, and the best configuration found is kept while the algorithm is searching for a better solution.

In the HC algorithm, the current configuration is always our best found so far.

In in the search process of TABU-search, the current configuration is not necessarily the best found so far. While the algorithm is searching for the best solution, the current may be a different configuration then our best found, and thats why the best configuration found is also saved.

In TABU search the *TABU-list* is used together with the greedy approach in order to select a next configuration from where the search is continued. The TABU-list contains all moves that can not be performed. It contains the k most recent performed moves, and it is used to prevent getting stuck in a local optimum. Because by this list does not allow to repeat the same moves over and over.

Below is the pseudo code of the Tabu search algorithm:

1: Let s be the initial configuration
2: Let TL be our *TABU-list*
3: Let b be our best configuration found so far
4: Let N count the amount of iterations the best configuration did not change
5: **while** $!MaxIterations$ **do**
6: Let $X(s)$ be the function that computes all configurations v that can be derived from s by doing a local change to s. Generate V, the set of permissible configurations, as following:

7: $V = \{v \in X(s) \text{ and } v \notin TL\}$

8: Using our evaluation function f, which computes the score of a certain configuration v, we compute the configuration in set V with the maximum score.

9: Choose from set V the $v \in V$ with the $max(f(v))$ and set $s = v$;

10: **if** $f(v) \geq f(b)$ **then**

11: $b = v$

12: $N = 0$

13: **else**

14: $N++$

15: break

16: **end if**

17: **end while**

18: **return** b

This algorithm starts with the initial configuration, which is a graph with or without edges (empty). In the case of a nonempty initial configuration the edges are either chosen randomly or based on our prior knowledge.

This algorithm stops when either the max amount of iterations is reached, or the best configuration so far has not changed for N iterations. This also prevents the algorithm going into an infinite loop.

In every iteration the set of *permissible configurations* V is computed, for every configuration inside V the score is computed using our selected evaluation function, in our pseudo-code denoted by function f. The current state s is then set to the state from set V with the highest score, $max(f(V))$. When the score of this current configuration s is higher then the best configuration found so far (variable b), the best configuration b is set to the current configuration s.

Chapter 5

Some experiments with synthetic datasets

In this part we execute experiments on the algorithms described in the previous section, in order to learn which algorithm performs best in creating a good network for our target dataset.

5.1 Background of the tests

We use three distinct datasets to test the performance of each algorithm. These datasets are created from three networks: the *SACHS*, *CHILD* and *INSURANCE* network containing respectively 11, 20 and 27 nodes. For simplicity, we refer to every dataset as the name of the network it is constructed from (e.g. SACHS network gives us the SACHS dataset), and we define these three networks as the *solution networks*, these are the networks that should be created from their randomly generated data.

All these networks are provided by the *Bnlearn package* [26] inside R. The networks inside the package are represented as *bn.fit* objects, these objects encode the structure of the Bayesian network and they also contain the probability distribution for every variable.

Every dataset is randomly constructed from the solution network, using a function called *rbn*. Using the *bn.fit* object, the *rbn* function can then randomly generate a dataset that should encode this network structure and probability distribution.

The first reason we selected these particular datasets to conduct our tests on, is because of the amount of variables they contain. These three datasets are similar in size compared to our target dataset, which contains 31 variables.

The other reason we use these particular datasets is that all three datasets contain only discrete variables, and they are all complete, just as our target dataset (from the algorithms perspective).

We use our selected algorithms to generate networks for these datasets, and we evaluate how good our algorithm has performed by comparing the generated network with the solution network. This comparison

is done by computing the *structural hamming distance* (short *shd*) between the two networks. The hamming distance simply gives us the number of permutations (adding/reversing/removing an edge) that have to be done in order to turn one of the networks into the other. We also take the average *log-likelihood* of the generated network given the data. By average is meant that we divide the log-likelihood of the entire network by the amount of records the dataset contains.

Adding prior knowledge to an algorithm can increase the performance of the algorithm. For our target dataset we have certain prior knowledge about the dataset, from this prior knowledge we can deduce certain relations between the variables of our domain. Inserting these relations into the algorithm can result in the algorithm creating a better network. When we create a network for our target dataset in section 7, we will also use our prior knowledge relations.

Adding prior knowledge to an algorithm may give us a different performance result for the algorithm. We demonstrate this in subsection 3 below.

In section 5.5 we make alterations to the parameters of the best performing algorithm of our test, to get an understanding of the influence of the parameters on the performance, and to find the parameter settings which work best for the algorithm.

All experiments are executed in R [27].

The algorithms that we test are:

1. The Grow-shrink algorithm (GS)
2. The Incremental association algorithm (IAMB)
3. The Hill climb algorithm (HC) using BIC score method
4. The Tabu search algorithm (TS) using BIC score method

5.2 Description of the algorithm tests

We have the SACHS dataset, containing 11 variables, the CHILD dataset containing 20 variables, and the INSURANCE dataset containing 27 variables. To test the performance, the steps below are followed. These steps are followed for one of our datasets D, using one of our selected algorithms A.

For every algorithm we initially use a range of records from 1000 to 10000 with step size 1000. Based on the results we then can change the range and step size in order better image of the performance of each algorithm.

1. Beginning with the dataset D
2. Randomly select N records from the existing dataset D to create our new dataset D_*
3. Using every of our algorithms A, a network G_A is created from this dataset D_*

4. For every network G_A the hamming distance between the solution network G_S and our network G_A and the log-likelihood of G_A given the dataset D_* are computed and stored
5. The steps from step 2 to step 4 are repeated 100 times
6. The amount of records N is then increased by 1000, and we go back to step 2, and we continue until we reach 10000 records
7. For every algorithm A, the average hamming distance and log-likelihood is computed given the amount of records N

In this way, we can see how good algorithm A performs in creating a network for dataset D containing N records, and we can compare the results the algorithms get.

5.3 Parameter settings

For every algorithm there are certain parameters that can be altered, to change certain aspects of the working of the algorithm. For every algorithm we use the initial parameter settings, these parameter settings are listed below.

5.3.1 Constraint based algorithm parameters

Both GS and IAMB have the same set of parameters, listed below:

- test: The label of the conditional independence test to be used in the algorithm. Value: The mutual information is used for categorical variables, the Jonckheere-Terpstra test for ordered factors and the linear correlation for continuous variables.
- alpha: A numeric value, the target nominal type I error rate. Value=0,05
- B: A positive integer, the number of permutations considered for each permutation test.
- max.sx: A positive integer, the maximum allowed size of the conditioning sets used in conditional independence tests. The default is that there is no limit on size.
- strict: A boolean value. If TRUE conflicting results in the learning process generate an error; otherwise they result in a warning.
- undirected: A boolean value. If TRUE no attempt will be made to determine the orientation of the arcs; the returned (undirected) graph will represent the underlying structure of the Bayesian network. Value=FALSE

5.3.2 Score based algorithm parameters

The following parameters are contained by both HC and TABU:

- max.iter: An integer, the maximum number of iterations. Value: infinite
- maxp: The maximum number of parents for a node. Value: infinite

Hc specific parameters:

- restart: The number of random restarts. Value: 0
- perturb: The number of attempts to randomly insert/remove/reverse an arc on every random restart. Value: NULL

Tabu specific parameters

- tabu: The length of the tabu-list used in the tabu function. value: 10
- max.tabu: The iterations tabu search can perform without improving the best network score. value: 10

5.4 Results

The results for each dataset can be found in figures below.

Sachs dataset

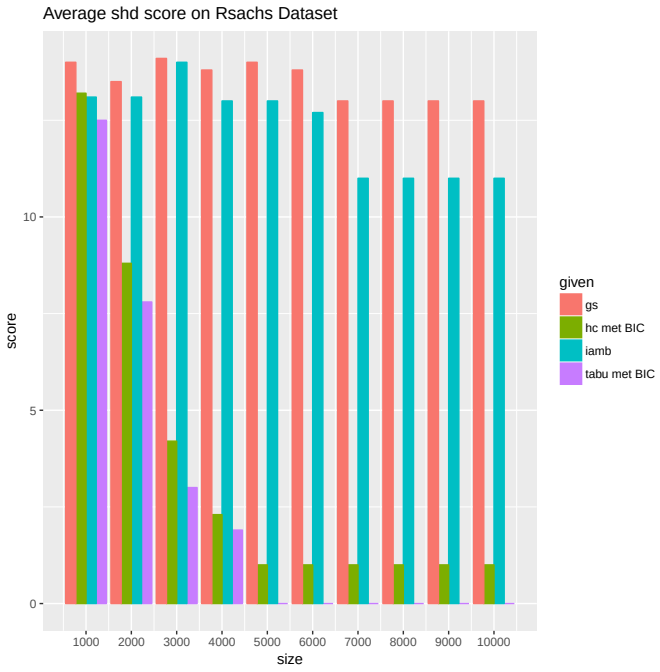


Figure 5.1: Results of the algorithms on the Sachs dataset

In figure 5.1 we can see that from $N = 5000$ on, the results slightly change (from $N = 7000$ no change).

To get a better insight in the performance we change the range of records to 500 to 5000, with a step size of 500 and we repeat the experiment. The result we get is found in figure 5.2 below.

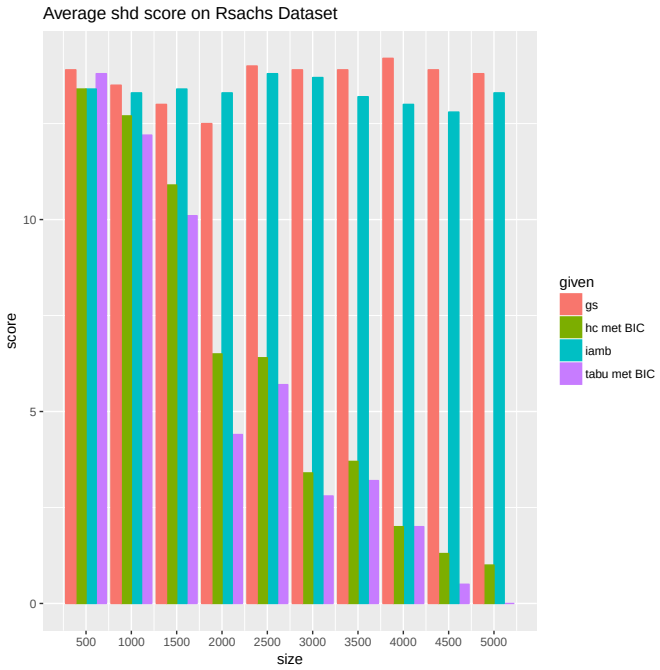


Figure 5.2: Results of the algorithms on the Sachs dataset

Child dataset

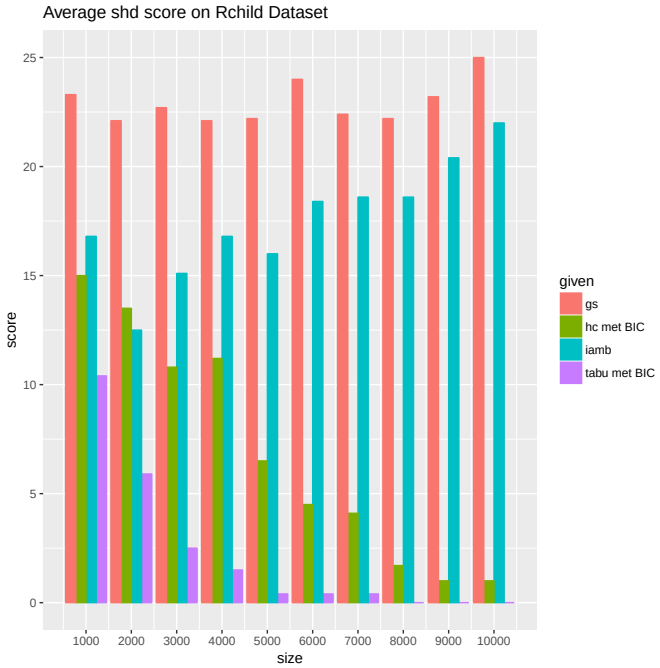


Figure 5.3: Results of the algorithms on the Child dataset

The performance of each algorithm is shown well enough by figure 5.3. No adjustment of the step size and

range is needed.

Insurance dataset

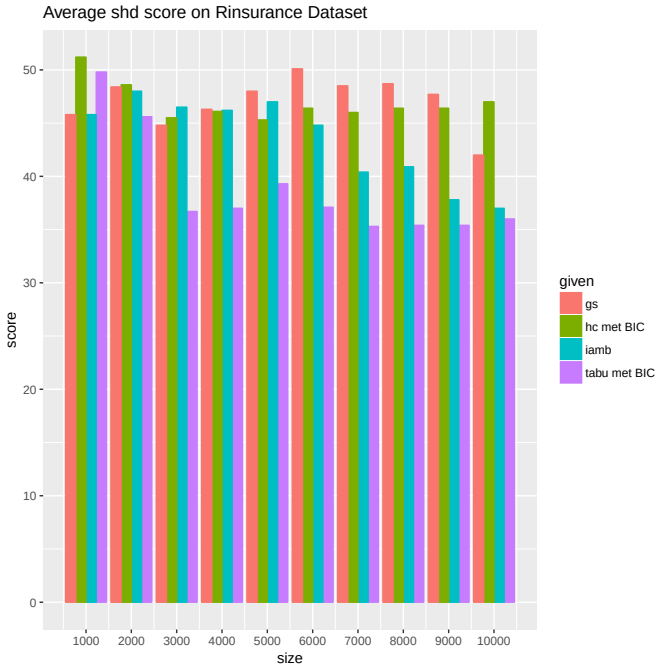


Figure 5.4: Results of the algorithms on the Insurance dataset

We see a slight trace of convergence in figure 5.4, so we expand the range to make sure the scores don't get better when more records are contained in the dataset. We adjust the range to 5000 to 50000 with step size 5000, and the new results are shown in figure 5.5 below.

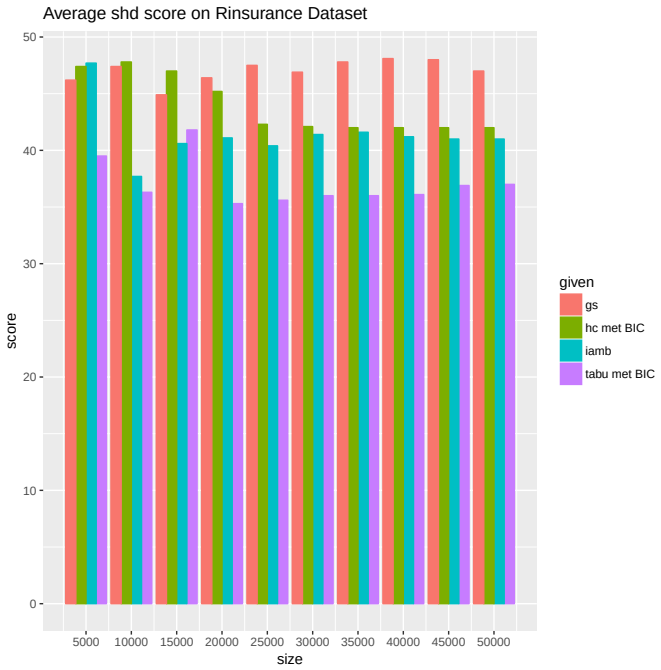


Figure 5.5: Results of the algorithms on the Insurance dataset

Using a wider range doesn't give us a better image of the algorithms performances.

5.5 Influence of added background knowledge on network score

Adding background knowledge to the network constructing algorithm can have a positive effect on the score of the constructed network. Below this is demonstrated using two of our selected algorithms, the Grow-shrink and IAMB algorithm. The dataset used here is randomly generated from the CHILD network and it contains 5000 records.

We start by constructing a network using the algorithm, and then we compare the constructed network with the CHILD solution network. We then select one edge that is present in the solution network but not in the constructed network, and that edge will be our prior knowledge when we run the algorithm again.

We use the parameter *white-list* to provide the prior knowledge relation to the algorithm. The relations in the white-list will always be contained in the resulting network structure.

As in the previous test, the *structural hamming distance* (shd) score is used to measure performance of the algorithm. But here, we also calculate the true-positive, false-positives and false-negatives between the solution network and the constructed network. We do this using the *compare* function.

Compare() gives us:

1. The number of true positives (short tp), this is the number of edges in current also present in target)
2. The number of false positives (short fp), this is the number of edges in current not present in target)

3. And the number of false negatives (short fn), this is the number of edges not in current but present in target)

The CHILD solution network is depicted in figure 5.6:

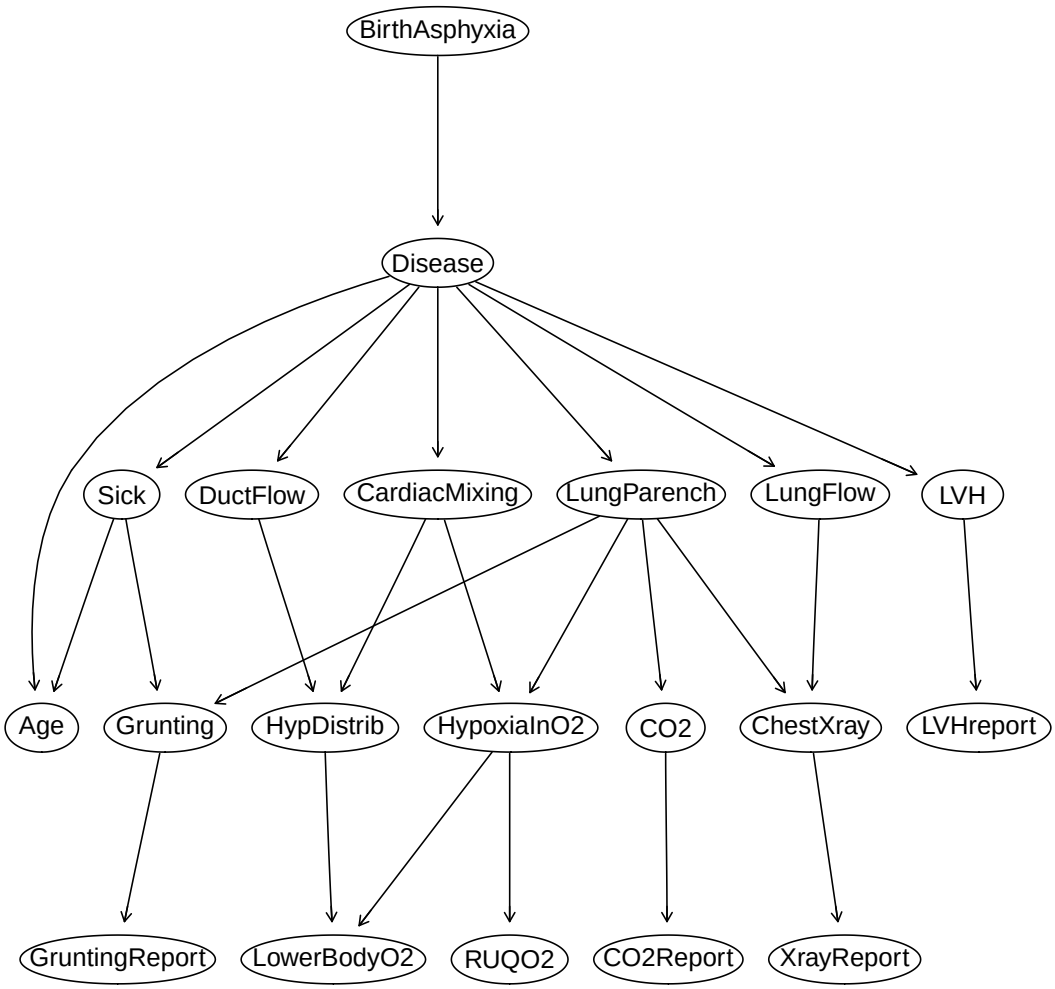
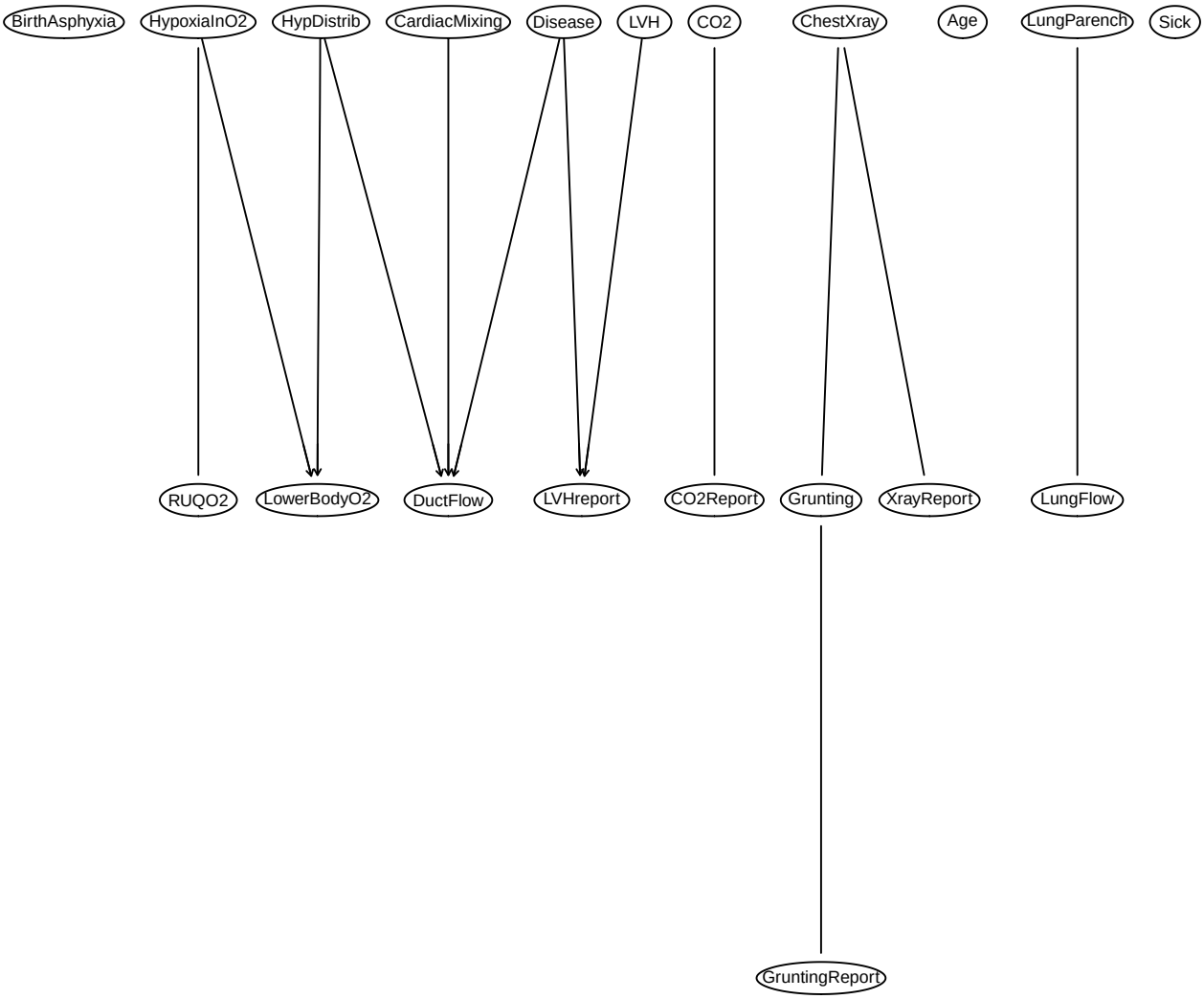


Figure 5.6: The CHILD network

When we construct network from the CHILD dataset using the GS algorithm we obtain the network structure and SHD results shown in figure 5.7.



shd score	26
tp	4
fp	9
fn	21

Figure 5.7: Network of GS without prior knowledge

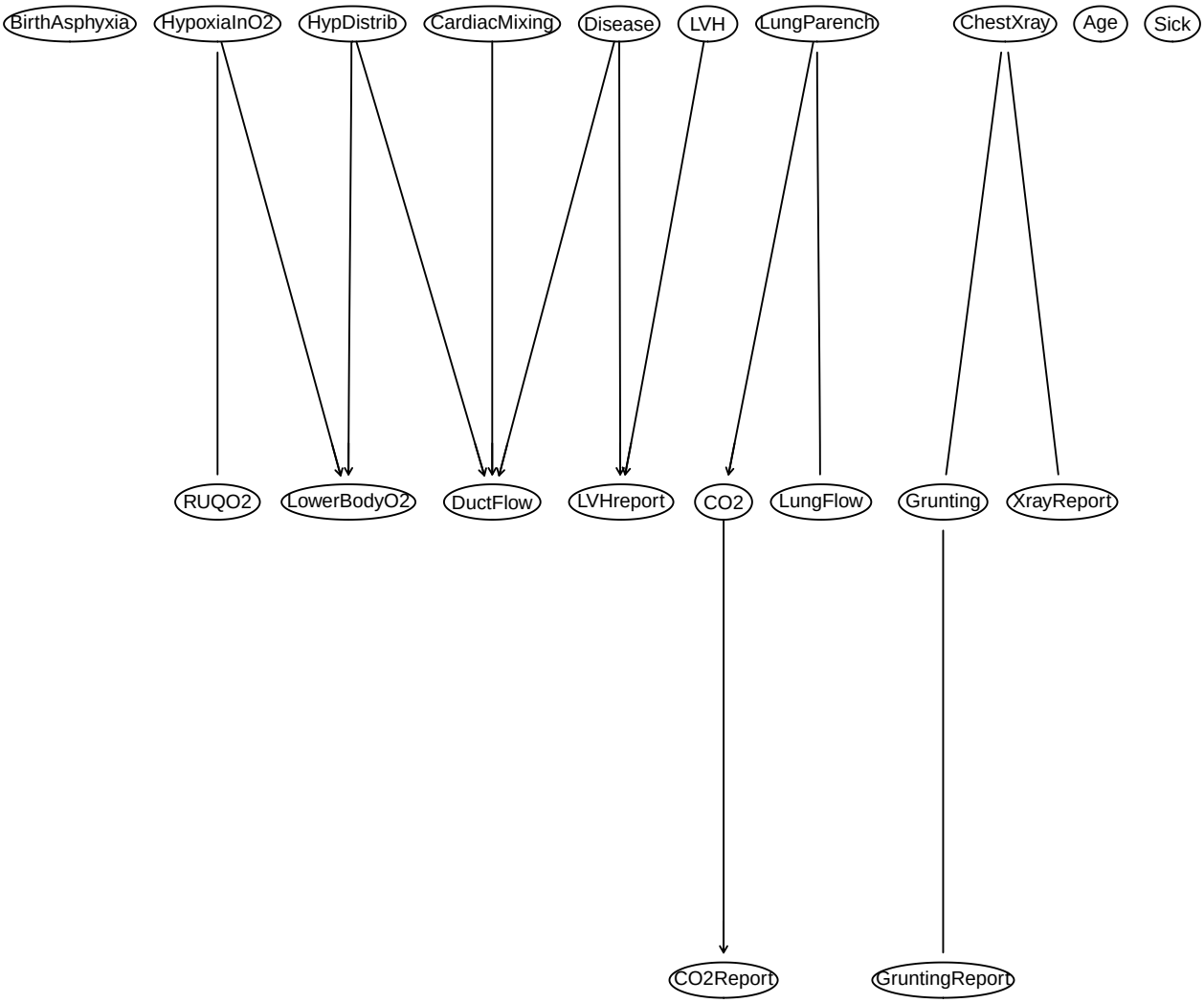
When we compare this network with the solution network, we can see that our constructed network contains far less edges then the solution network, and that the edges the constructed network contains are mostly undirected.

One positive remark on the constructed network is that most of the undirected edges it contains are present in the target network, only in the target network they have the right direction.

Now, consider the variables *CO2* and *CO2Report* in our constructed network. We can see that there is an undirected edge between these two variables, and that both variables are unconnected to the rest of the variables in the network.

When we take a look at these two variables in the solution network, we can see that they are connected by an directed edge from *CO2* to *CO2Report*, and we see that *CO2* has an incoming directed edge from variable *LungParench*.

Next, we repeat creating a network using GS, only this time we add the directed edge from *LungParench* to *CO2* to the white-list. The network structure we obtain and SHD results are shown in figure 5.8.



shd score	25
tp	6
fp	8
fn	19

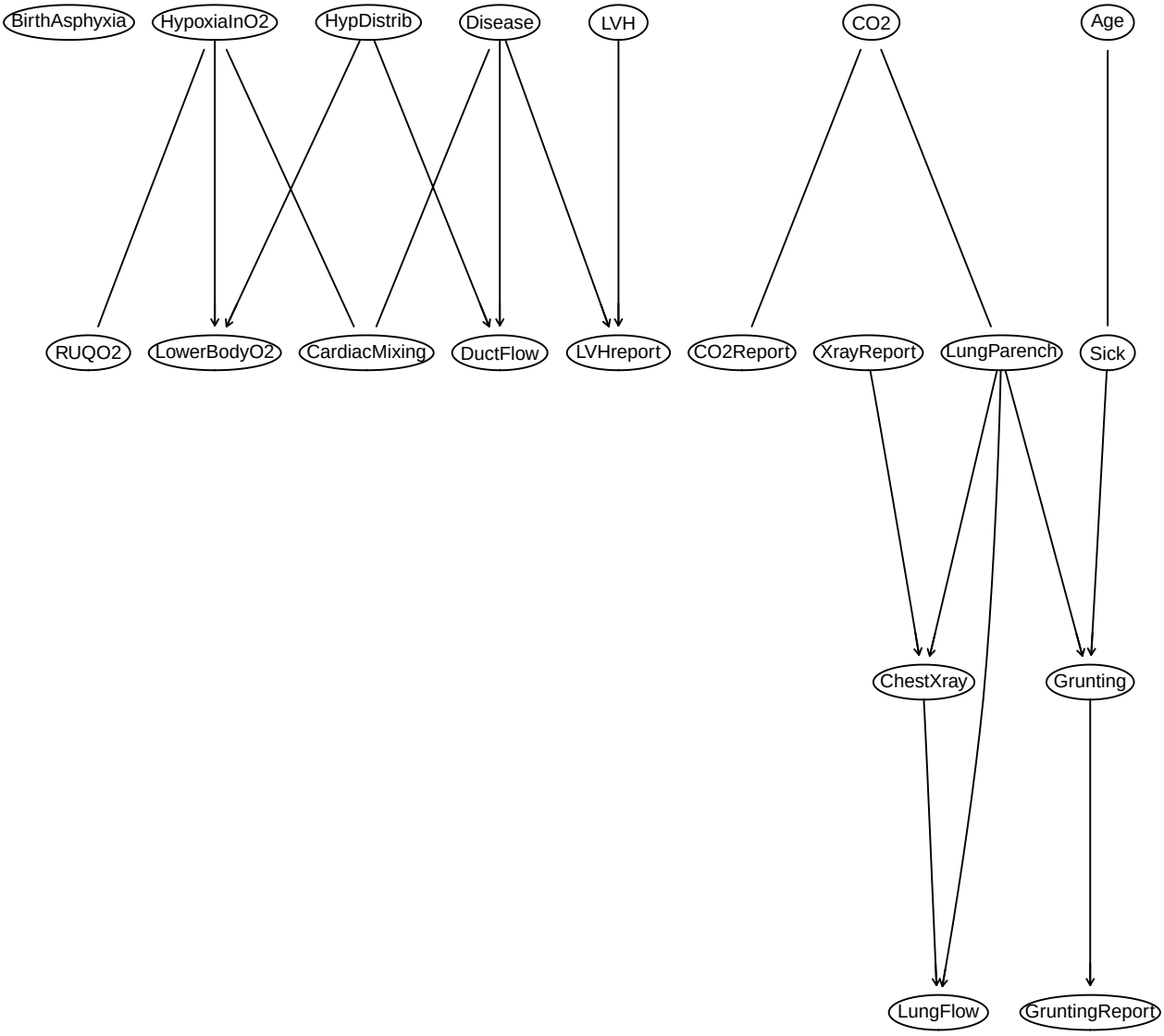
Figure 5.8: Network of GS with one prior knowledge relation

When comparing the shd of the network of figure 5.7 with our current network in figure 5.8 we see a decrease in the shd score, and an increase in the true positives (from 4 to 6) and both a decrease in the false positives and false negatives.

Looking at the current network we see that variables *CO2* and *CO2Report* are now connected by a directed edge from *CO2* to *CO2Report*, exactly as in the target network. By this we see that adding prior knowledge can

result in the network getting a better score.

With the IAMB algorithm we obtain the network structure and SHD results shown in figure 5.9.



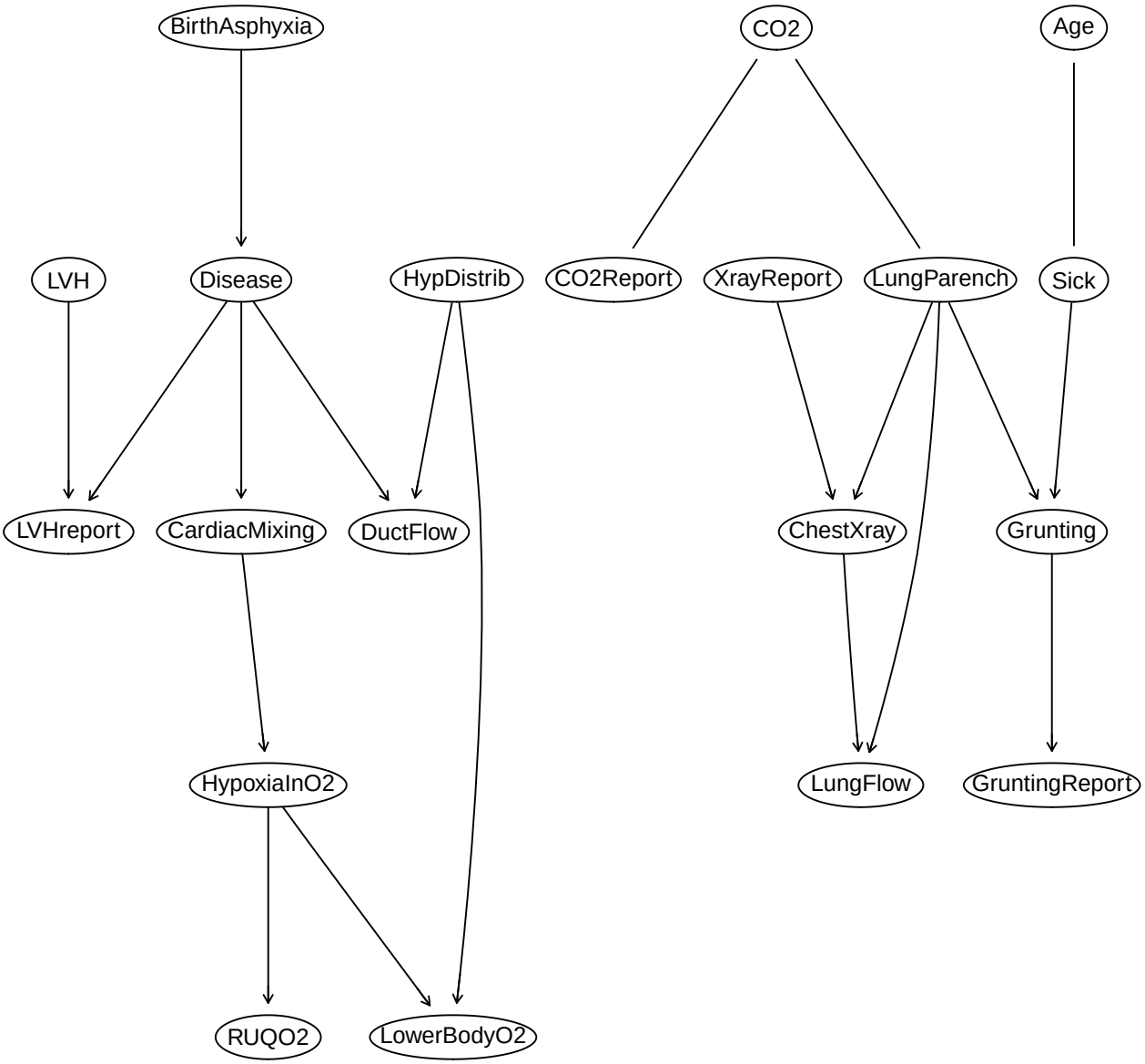
shd score	17
tp	8
fp	11
fn	17

Figure 5.9: Network of IAMB without prior knowledge

One of the missing relations in the above network is the edge between *BirthAsphyxia* and *Disease*.

We now repeat constructing a network, only with this relation added to the white-list. We obtain the network

structure and SHD results shown in figure 5.10:



shd score	16
tp	12
fp	8
fn	13

Figure 5.10: Network of IAMB with one prior knowledge relation

Comparing these results to the previous we can see that the addition of prior knowledge resulted in a better network. The true-positives increased by 4, and the false-negatives and false-positives decreased by 4 and 3 respectively. While the shd score only decreased by one, we can say the network got better.

When comparing the above network to the previous, we can see that in the previous network the edges

between *Disease* and *CardiacMixing* and *Disease* and *DuctFlow* had no direction, and in the above network they have the right direction. The addition of prior knowledge thus resulted in both these edges getting the right direction.

5.6 TABU parameter test

For these tests we use a randomly generated dataset of the CHILD network with 1500 records. The same size as our target dataset.

We alter the value of two parameters of TABU in order get an insight in which value gives us the best performance. The performance is again based on the shd between the CHILD solution network and generated network.

The two parameters are the tabu and max.tabu parameter.

With parameter tabu we can change the length of the TABU list. The initial value is 10. With max.tabu the maximum amount of iterations TABU search can perform without improving the best network score can be set. The initial value of this is also 10. The value set for max.tabu must be smaller or equal to tabu (because the TABU-list is already full when they are equal).

We run TABU altering the value of these parameters from 1 to 20 with step size 1. Our results are in the bar-chart in figure 5.11.

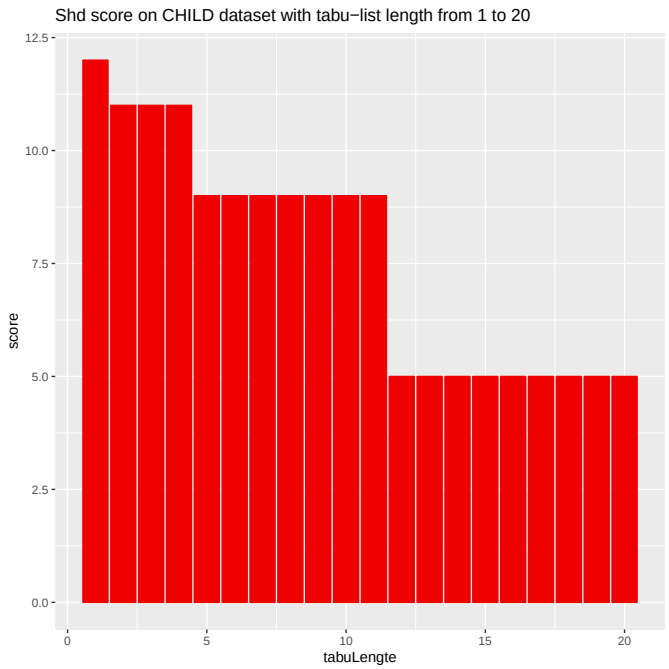


Figure 5.11: Results of varying the TABU-list size using TABU on CHILD data

In the results we can see that for a TABU-list of length 12 or more, the algorithm learns the network best

(shd=5). Increasing the TABU-list size above size 12 does not give us a better performance, but also no decrease in score (tabu=100 gives us a shd of 5).

So it is important to give the TABU algorithm a large enough TABU-list in order to guarantee an optimal performance of TABU.

5.7 Conclusions

We start by discussing the results of the algorithms on the SACHS dataset.

In the first bar chart with the wider range (figure 5.1) we can see that, at 1000 records, the results of the algorithms are somewhat close to each other, but going further we can clearly see that the score based algorithms outperform the constraint based algorithms by far. The TABU algorithm performs best. Looking at the next bar chart with the smaller range (figure 5.2), we can see that TABU outperforms the rest of the algorithms, needing 5000 records to completely learn the network.

Moving on to the CHILD dataset (figure 5.3), we can see that TABU again only needs around 5000 records to completely learn the network. We can also see that TABU performs very well with a small data-size. At 1000 and 2000 records its score is significantly better then the other algorithms scores.

Looking at the results of the INSURANCE dataset (figure 5.4 and 5.5), we can see that of the three datasets, this is the hardest dataset to learn for our algorithms. The results at 1000 and 2000 records are very close to each other for all algorithms, but we can see that when the data-size increases, the TABU outperforms the other algorithms.

Now when we consider our target data, the gynecological dataset, we know that it contains slightly more then 1500 records and 31 variables. Looking at our test results we can see this is a small amount of records for a dataset containing 31 variables.

In order to learn the best network for our target dataset, we have to select the algorithm that learns the network with the smallest data-size possible. On the other hand it must also be capable of creating a good network from a dataset with a small data-size.

Compared to the other algorithms, TABU performs best on these areas.

Chapter 6

Description of the medical problem domain

Ovarian cancer is a disease where a malignant tumor forms in one or both ovaries. There are three types of tumors that can form in the ovary:

1. Epithelial tumors, which arise from the epithelial cells.
2. Stromal tumors, which begin in the ovarian tissue that contains hormone-producing cells.
3. Germ cell tumors, which begin in the egg-producing cells.

Most malignant tumors that form in the ovary arise from the epithelial cells (more than 90%), these are called *ovarian carcinoma*. In the next part we use data to get more insight in the cause and treatment of ovarian carcinoma.

The cells of the ovarian carcinoma have several features, which are used to classify them. The *Serous carcinomas* are the most common ovarian cancers; they often have the form of cysts which includes a clear fluid (see figure 6.1). Other common types are the mucinous and endometroid.

Figure 6.1: Ovary, cut in two halves with a serous cystadenocarcinoma.

6.1 Symptoms and diagnosis

Most common ovarian cancer symptoms are abdominal and pelvic pain and abdominal enlargement due to the large size of the tumor. Other symptoms are constipation and bladder problems, nausea, fatigue.

The problem is, these symptoms are in most cases not apparent until in the disease is in a late stage. One of the reasons of this absence of symptoms is the space around the ovaries. Because of this the tumor can silently grow in this space.

Thats why diagnosis of the ovarian carcinoma is ofter very late, at a stage where the disease has spread through the body.

Staging is used to determine how far the disease has progressed. Things that play a role in determining the stage of the disease is how large the tumor has grown and how aggressive it is, and whether the disease has spread trough the body.

These stages are defined by roman literals. In stage I the disease has least progressed, and in stage IV the most.

These stages then determine the form of treatment that is given to the patient.

As noted before, many diagnosed cases of ovarian carcinoma are late stages. In 60 to 70% of the diagnosed patients, stages are between II and IV. Ovarian carcinoma is therefore known as a silent killer

6.2 Treatment

There are a number of different steps in the treatment and management of the ovarian carcinoma. Simply put, there are two large categories: *surgery* and *chemotherapy*. In surgery a gynecologist or oncological-gynecologist will carry out surgery on the patient to remove as much of the malignant mass as possible. In chemotherapy chemotherapeutic drugs are used. Most patients with ovarian cancer get a combination of both surgery and chemotherapy.

When the original stage was favorable and the tumor was diagnosed at an early stage in its development, follow-up examinations serve the purpose of early diagnosis of recurrence.

6.3 Survival

As stated above, stage of the cancer determines the survival rate. It is shown that several factors play a role in the survival rate, examples are age and personal condition.

The five year survival rate is used to determine how many patients have survived for 5 years after the treatment. This rate shows that patients that are treated for a cancer in early stage, have a much higher survival rate then patients with cancer in a later stage.

The graph in figure 6.2 below shows the survival rate in percentages in the Netherlands [28].

For every 10 year time frame is shown what percentage of patients have survived after a certain amount of years after diagnose.

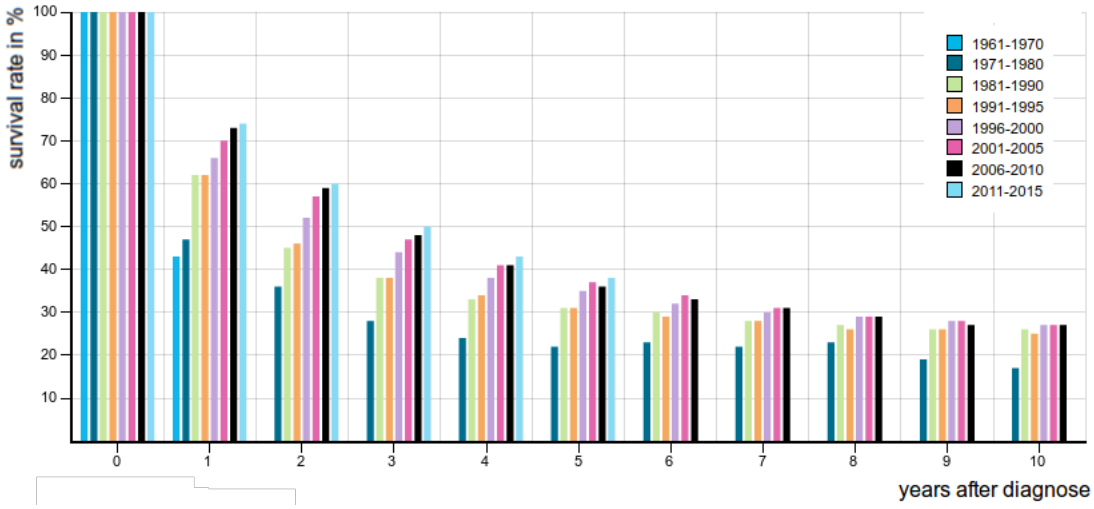


Figure 6.2: Survival rate ovarian cancer in the Netherlands

6.4 Motives for building a BN for our medical data

- About 5% of the detected malignant tumors in the Netherlands in females are ovarian cancer. On a yearly basis, 1300 new females are diagnosed with this disease.
- As also stated above, the ovarian carcinoma has a very bad prognosis and therefore, in contrast to its relative scarcity, is responsible for 50% of all gynecological cancer mortality. In the Netherlands, 1000 females die annually from this disease.
- At around 5% to 15% of all women with ovarian carcinoma its hereditary.
- The actual cause of this cancer is not clear. But there are factors that affect the probability of getting ovarian carcinoma. One big factor that affects this probability is the number of ovulations and/or children a patient has had.
- Only one fourth of the cases is detected at an early stage; in 60% of the cases the cancer has already been spread through the body. The reason for this is that the cancer develops in the abdomen without special special signs or symptoms.

By learning a Bayesian network from the medical data, the relations between relevant variables in the domain can be mapped and causal relations can be found between them. Bayesian networks can be used to learn the

causes of this type of cancer. The more causes are known about this disease, the more people can be diagnosed with the disease. In this way it more people can be diagnosed in an early stage. As stated above, it is important to diagnose ovarian cancer in an early stage. The earlier the cancer is diagnosed, the higher the chance the patient will survive the cancer. Thus more people can be diagnosed in an early stage, if causes are better understood.

Using Bayesian networks the influence of every treatment can be mapped better. The better every treatment is understood, and the effect on every type of patient (e.g early stage, later stage), the better a fitting treatment can be selected.

Thus, our aim with building a Bayesian network for our target data is to get insight in how these ovarian carcinomas are caused and can be treated.

Chapter 7

Learning the network from the gynecology dataset

We will build the network of the dataset described below.

7.1 The dataset

Our dataset, *ovariancancer*, contains 31 variables and 1555 records. The dataset we use is constructed from two datasets, obtained from two epidemiologists [25].

These two datasets contain information about ovarian cancer patients in the Netherlands. The datasets contain different variables ordered categorical wise. Both datasets are discrete, but incomplete.

We started by combining these two datasets together and in that way we got a complete joint of the two datasets.

From this joint dataset, we formed a new dataset by selecting 31 variables.

We used imputation to make this dataset complete. We use the function *impute* with *mean* inside the *Hmisc* package [23] to do this.

This function simply does the following. The missing values of every variable, are replaced with the value of that variables with the highest frequency.

In section 9 the list of selected variables can be found, ordered by category.

We have certain prior knowledge about this dataset. Part of this knowledge is our domain knowledge of ovarian cancer. We use this knowledge in order to learn a better causal network for our dataset. Using the blacklist and white-list we insert our domain knowledge into the network constructing algorithm.

Below we will first construct the network of the dataset with our selected algorithm and parameter settings without adding prior knowledge.

We will then analyze the outcome.

To get a better idea about how good the network is, we can compare the log-likelihood of our network with networks constructed from other algorithms. Good algorithms to compare with are the *Naive Bayes* algorithm and the *TAN* algorithm. Both algorithms build Bayesian networks from data using classification. On beforehand, one node has to be selected, called the class-node.

In Naive Bayes, the class node has outgoing arrow to all variables (attributes).

In TAN, the class node also has outgoing arrows to all other variables. Only in TAN, the variables can also have (at most) one incoming edge from an variable other than the class node. A more elaborate description of the functionality of these two algorithms can be found in the 1997 work of N. Friedman [24].

On beforehand, we know that both the TAN and naive Bayes should have a worse log-likelihood compared to our used algorithm, because both algorithms create network structures that are not learned from data. While both methods are proven to create good classifiers [24].

Thats why it is good to compare the log-likelihood of our network with our constructed network.

After we analyzed the network we add our domain knowledge to the algorithm and repeat constructing a network. In order to compare the network generated with domain knowledge with the network created without, we will also take the shd between the two.

We construct a Bayesian network for the ovarian cancer dataset using the TABU algorithm with BIC. All variables in the dataset will be represented by nodes in the network.

We use the following parameter settings.

- max.iter and maxp set to infinite
- tabu and max.tabu set to 20 to ensure our TABU-list is large enough

7.2 Network construction using TABU

Using TABU with BIC and our described parameter setting to build the network of our data we get the network shown in figure 7.1.

In table 7.1 the log-likelihood is given of the network of Figure 7.1, and the network we get from Naive Bayes and TAN with both the variable 'FiveYearSurv' as class-node. Here the log-likelihood is calculated for the entire network, and divided by the amount of records in the dataset.

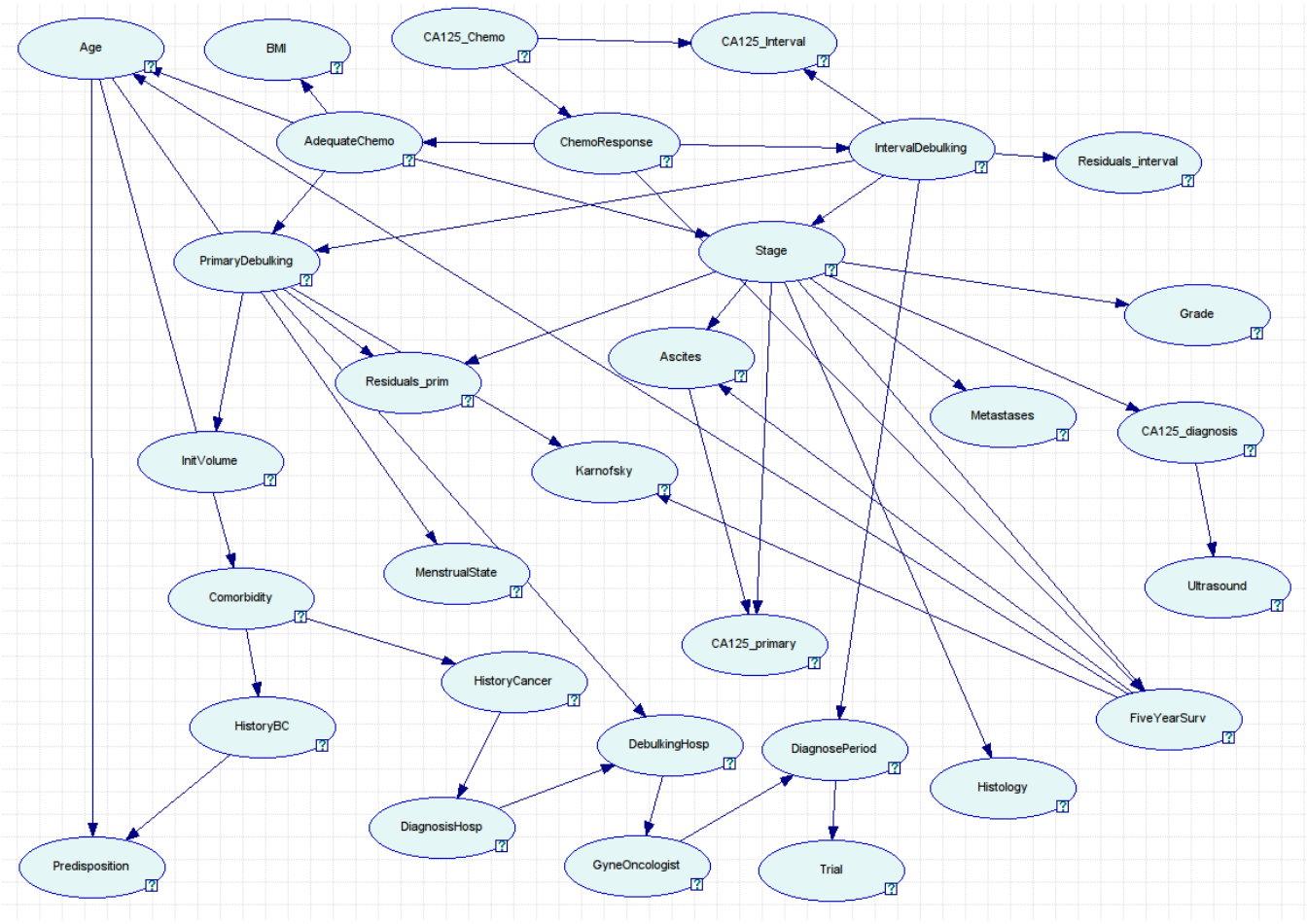


Figure 7.1: Bayesian network learned from the gynecological data using the TABU search structure learning algorithm

Algorithm	log-likelihood
TABU	-17.17673
Naive Bayes	-21.4298
TAN	-17.22865

Table 7.1: Comparison of log-likelihood of various Bayesian network learning algorithms

We can see that the TABU algorithm has the best log-likelihood score. This is in accordance to what we expected about the performance of TABU.

Now we use our domain knowledge to analyze the network. We can conclude that the following is wrong with this network:

- First of all, we have node FiveYearSurv in the network, which corresponds with the variable indicating the five-year survival rate. This node must not have outgoing edges, because we only want to see how other variables influence this particular variable.
- The following edges have the wrong direction:
  CA125Chemo to Chemoresponse

IntervalDebulking to PrimaryDebulking
FiveYearSurv to Karnofski
DebulkingHosp to GyneOncologist

We repeat generating a network with the background knowledge added. To make sure FiveYearSurv has no outgoing edges, we blacklist all edges that go from FiveYearSurv to any other node, and all (four) above listed edges with the wrong direction are also put in the blacklist, notice that we did not had to place edge FiveYearSurv to Karnofski in the blacklist, this is because we already put it in it in order to make FiveYearSurv have no outgoing edges.

We choose to only blacklist the edges with the wrong direction, instead of also white-listing them. The reason we do this is that we do not want to base the generation of our network on the edges we corrected. So we only want to make sure that the algorithm does not add the edges with the wrong direction to the network.

The only thing that we concluded from our background knowledge is that, the particular edges in the current network configuration have not the right direction. So we do not know for sure whether have the particular edge with a changed direction is the best solution.

In this way we get the network shown in figure 7.2.

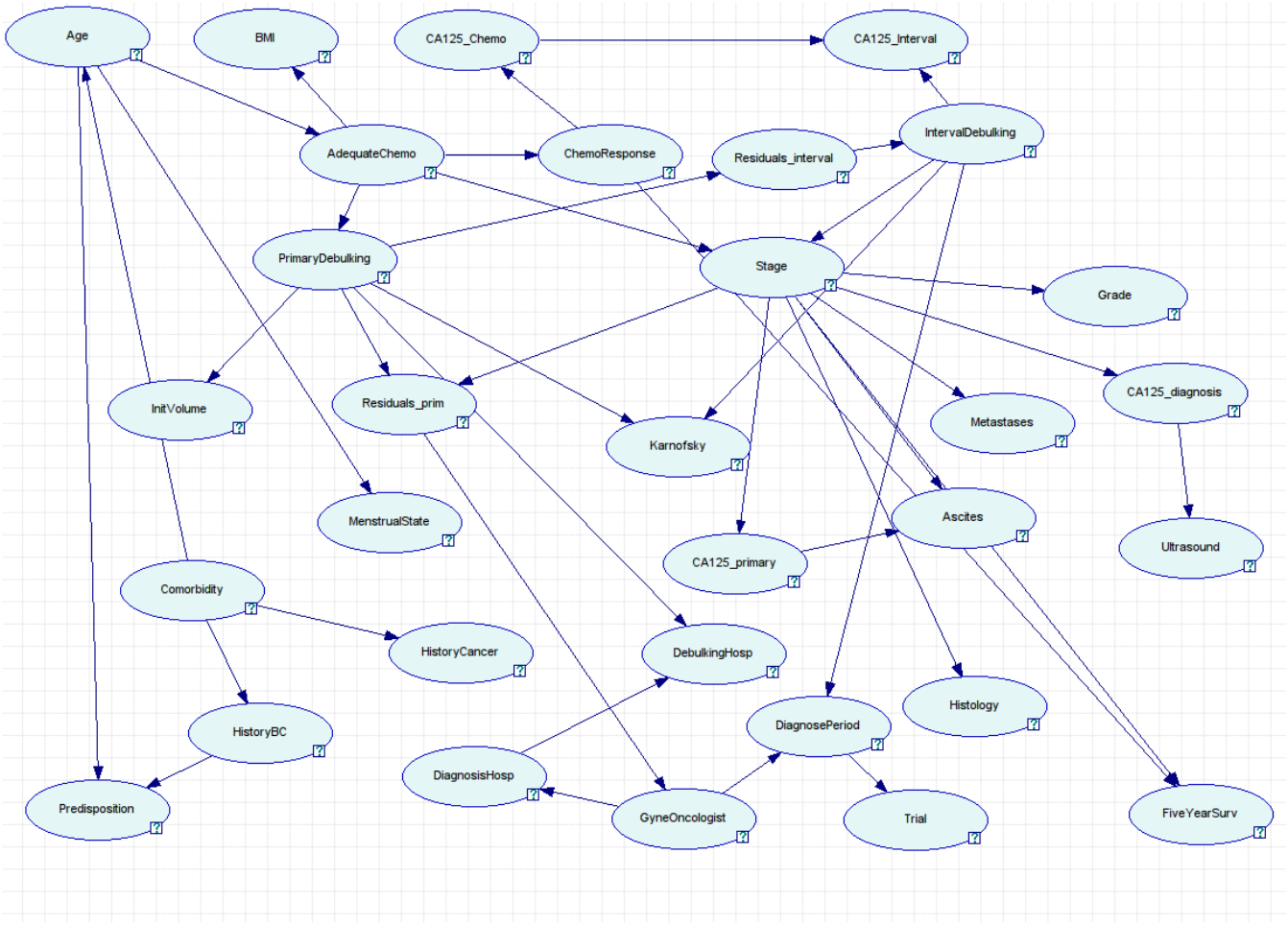


Figure 7.2: Network learned using background knowledge

Log-likelihood network	-17.29853
Shd between this network and previous	18

Table 7.2: Log-Likelihood and shd of network in figure 7.2

We can say the following about the changed network structure without invoking our domain knowledge.

First of all, the average log-likelihood of this network structure is little higher then the previous (now -17.29853 and first -17.17673), and looking at the shd between both network we can see that the addition of our knowledge resulted in a significant amount of changes.

We can clearly see that the FiveYearSurv node has no outgoing edges.

Looking at the edges which we wanted to change the direction of, we can see that edge CA125Chemo to Chemoresponse has changed direction.

The other three edges from the old network, IntervalDebulking to PrimaryDebulking, FiveYearSurv to Karnofski and DebulkingHosp to Gynecologist have not changed direction and are not contained in our new network structure.

But looking at their place in the network we can see that their place has changed.

- IntervalDebulking was the parent of PrimaryDebulking: IntervalDebulking is now the grandchild of PrimaryDebulking
- IntervalDebulking was the parent of PrimaryDebulking: IntervalDebulking is now the grandchild of PrimaryDebulking
- FiveYearSurv was the parent of Karnofski These nodes are no longer connected, and both are no decendants or ancestors of each other.
- DebulkingHosp was the parent of GyneOncologist DebulkingHosp is now the grandchild of GyneOncologist.

This is exactly what we intended with only blacklisting the edges with wrong direction. The only knowledge we had is that the edges had the wrong direction. Now we see that in 2 of our 4 cases, the algorithm placed the parent of the blacklisted relation as a grandchild, so still it had changed the direction only it has placed a node in between. Then there was one case the direction changed, and in the other the there is no directed path between the two nodes anymore.

Now we analyze the network structure again using our domain knowledge, to conclude how much better these changes that we made are compared to our previous network.

First of all, we can say that from our domain knowledge, the network with background knowledge is the better network compared to the previous constructed network without background knowledge. Besides the fact that all wrong relations are no longer present in the new network, also the structure of the network with background knowledge is in accordance to what we know about the domain. So after analyzing we can conclude that our new network is the better network.

And we can conclude that there are no changes that have to be made to the network structure in figure 7.2 . So this is our final network structure for the ovariancarcinoma dataset.

7.3 Clinical use of the ovarian cancer Bayesian network

In this section is shown how a Bayesian network on ovarian cancer can clinically be used. In contrast to the more usual probabilistic methods in medicine that are typically based on regression, e.g., logistic regression, we are not dealing with a model that only contains one random variable, used as outcome variable, where all other variable are deterministic. Such typical medical probabilistic models can only be used in *one* way: given values for *all* input (predictor, or independent) variables, one is able to predict the value of the outcome or response variable. In a Bayesian network all variables are random variables as this means that a Bayesian network can be used in multiple ways, i.e., each of the (group of) variables can be predicted as response variables, where in addition each variable can act as input. It is not necessary that all non-response variables

are known: a Bayesian network is perfectly capable to deal with unknown values, taking into account the values of variables that are known. This opens up multiple ways to use a Bayesian network in a clinical context.

7.3.1 Prior Bayesian network

The first useful way to apply a Bayesian network is as a summary of the data on which it is based. Not only does it shown the dependence and independence information between the variables included in the Bayesian network, it also is able to shown the marginal prior probability distribution. These distributions give a good summary of the population of patients on which the learning (both structure and parameter) has been based. Figure 7.3 gives such a summary of the Bayesian network previously shown in figure 7.1.

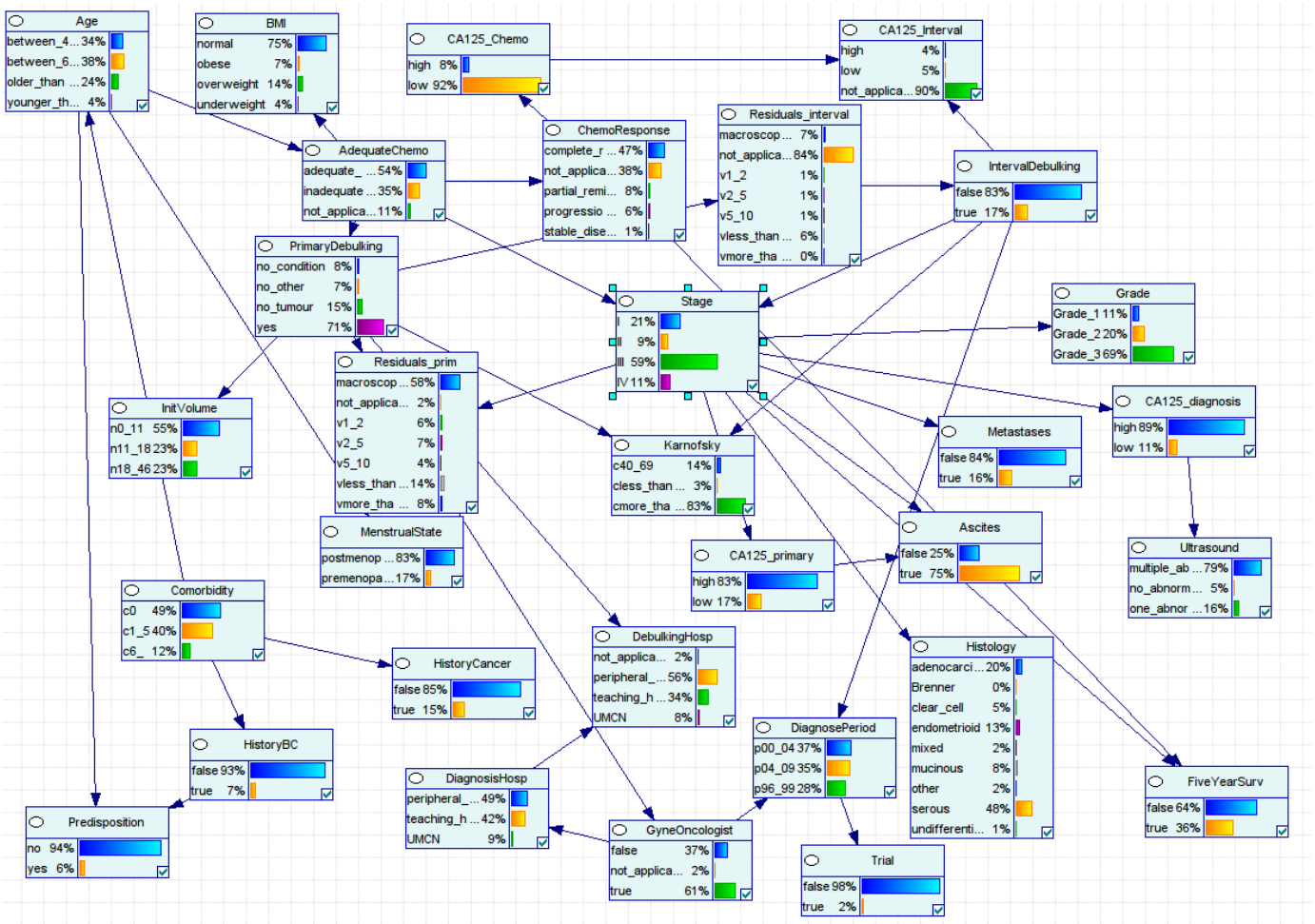


Figure 7.3: Marginal prior probability distribution of the Bayesian network shown in Figure 7.2

The above prior probability distributions shows us that for the entire population only 36% survive more than 5 years and that the majority of patients (69%) has a grade 3 type of cancer. So that does not look good. However, on the positive side only 16% of the patients have metastases. Clearly, quite a lot of useful information can be extracted from the prior distribution.

7.3.2 Prognostic use of the Bayesian network on ovarian cancer

The most useful way in which Bayesian network on ovarian cancer can be used is for the prediction of the five-year survival for specific patients. So basically, one computes

$$P(\text{FiveYearSurv} \mid E)$$

where E is all the patients evidence that is available of a specific patient, and which may concern a subset of all the variables in the model.

This is what is done for the Bayesian network shown in figure 7.4. Note that for this specific patient the five-year survival goes up from 36% (as shown in figure 7.3 to 81%. So clearly, these patient findings have a major positive impact on the survival.

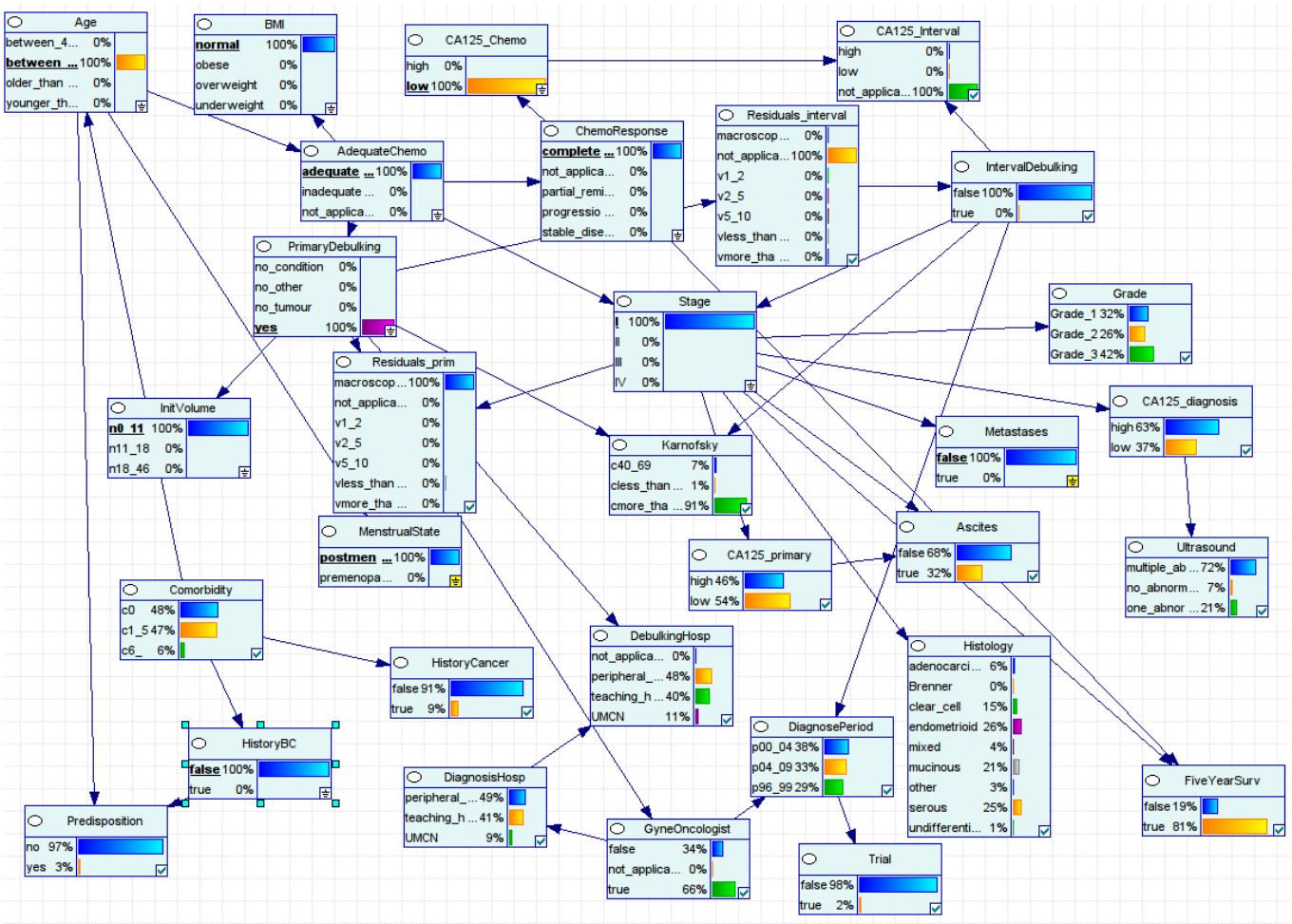


Figure 7.4: Posterior probability distributions of the Bayesian network shown in Figure 7.3 with patient evidence entered into the network

7.3.3 Subpopulation description

The final typical use of a Bayesian network for ovarian cancer is to use it to understand the composition of a particular patient population by fixing some of the variables to a particular value.

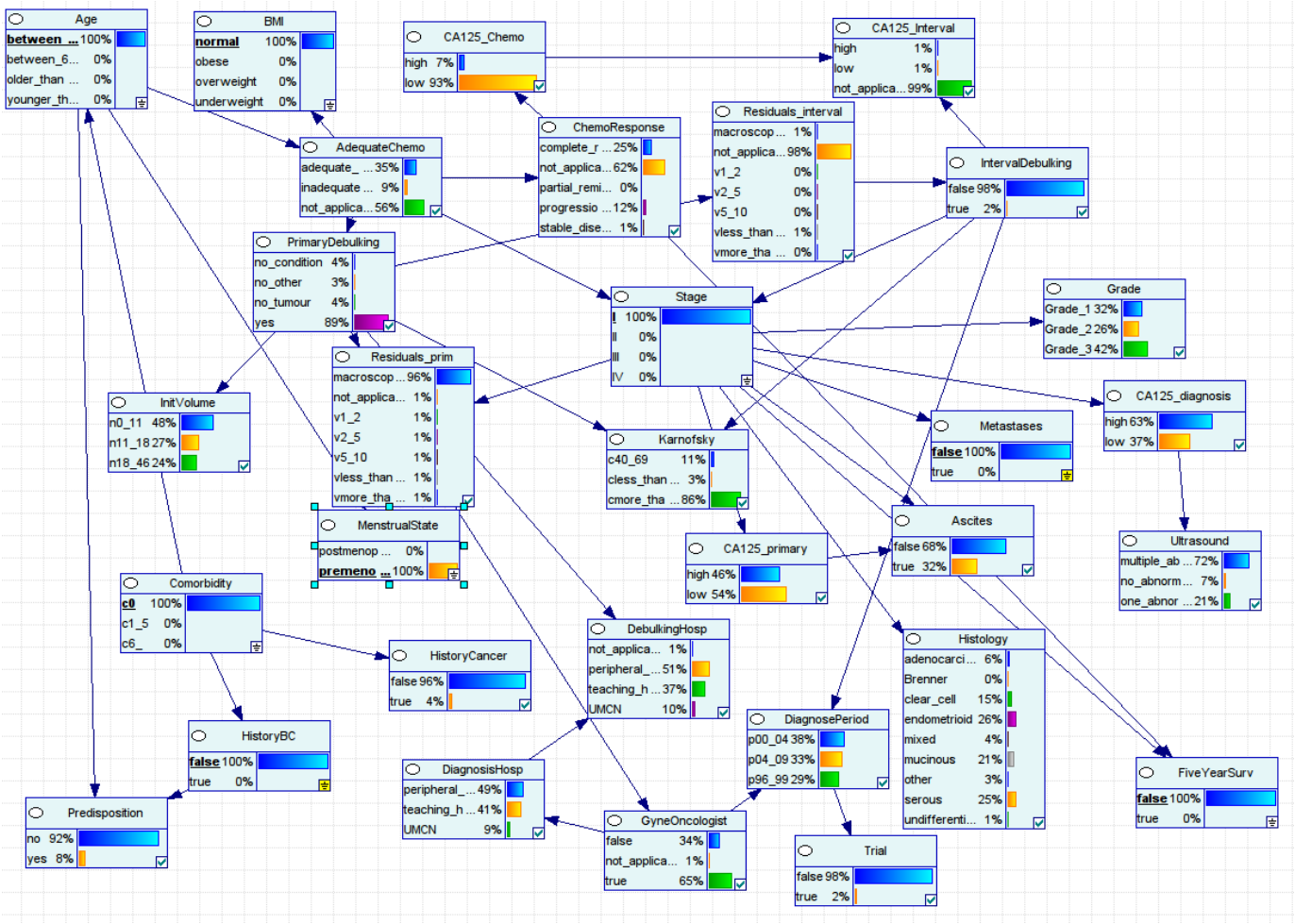


Figure 7.5: Posterior probability distributions of the Bayesian network shown in Figure 7.3 used to describe a subpopulation of patients

In Figure 7.5 is shown a Bayesian network that is meant to consider the subpopulation of patient whom have not survived five years, but have the favorable Grade 1, no metastases, have normal BMI, age between 40 and 60 years, premenopausal, and no comorbidities. The consequence of such characteristics can be read off from the Bayesian network. For example, these patients have primary debulking (the tumor is removed entirely) and no chemotherapy or adequate chemotherapy. In other words, the treatment of these patient have been particularly mild, but in the end it was not effective.

Chapter 8

Conclusion and future work

We have seen that the TABU algorithm appears to be good enough for building a good Bayesian network for our target data. As discussed in the previous section, the likelihood was better when compared with other algorithms. Following an analysis of the network constructed by TABU, we were able to build an even better network by incorporating domain knowledge. Figure 7.1 shows the result of structure learning only, whereas figure 7.2 display the result of learning after incorporating background knowledge. This last network (the one with background knowledge) had a slightly worse likelihood compared to the one without background knowledge.

We can remark that the best network is not necessarily the network with the best likelihood. The likelihood score tells us how well the network fits the data, and surely, that does not have to be the best network structure for the variables in the domain. That is because datasets have their limitations. Not all relations found in the data are in fact relations in the real world domain. That is why domain knowledge must be used to determine how reliable the found relations are.

There are several things that may have led to our data being less representative for the variables in our domain. First our dataset size, the amount of records, may have been too small for the amount of variables it contains.

In the tests of section 5 we have shown that 1500 records is too small to learn the complete network structure. For example, the CHILD network with 20 variables could be learned well using tabu, but it needed at least 5000 records to learn (roughly) the complete structure. A data-size of 1500 records was too small to learn the complete structure, and it performed much worse.

Our dataset has more variables then the CHILD dataset, and this tells us that 1500 records is probably too small to learn the exact network structure.

Also, the data we used was initially not complete. We had to make use of imputation to make it complete, and that can also influence how representative the data is for the real domain.

We made use of a simple imputation method, where the value with the highest frequency was used. This

may have led to our data being less representative. In our case we chose to put less focus on the imputation method, mainly because we decided to focus on the understanding and using of the network constructing algorithms. Thats why this can be a good part of future work, for example to compare existing imputation methods performances in search for the best.

As we can see, the best network is not necessarily the network with the best likelihood, the domain knowledge also plays a big part in determining how good the structure is. The likelihood only tells us how well the structure fits the data, and not how well it fits the actual real world domain. The final Bayesian network we build can be used to make predictions about treatment and lab results for patients.

Only, as already mentioned above and further mentioned below when we speak about future work, there are several things that can be done that may lead to a better network structure. And of course, the most preferable would be to have the best network structure possible when predictions have to be done for patients.

Knowing this, one should be very careful in drawing conclusions solely based on the network. Conclusions should only be drawn by someone with also the domain knowledge, a specialist in the domain of the data.

And even if used by one with domain knowledge, the results have to be taken with care. Because, as mentioned, the data itself has his shortcomings.

So to conclude, we were able to construct a good causal Bayesian network using TABU, but it should be taken with care when used.

In this thesis we compared the performance of four BN constructing algorithm in order to find the best. As mentioned in section 4, the reason we used four is mainly time as a limiting factor. Using four algorithms we could compare two of each category, while still cover the functionality of each algorithm in reasonable depth.

Of course, it is possible that there are algorithms that perform better in creating a network for our target data. This can either be a variant of the tabu algorithm that we used, or a completely other algorithm.

That's why one possibility of future work is to compare the performance of more algorithms in search for a better algorithm.

The TABU search algorithm we used in section 7 is from the Bnlearn package. We did alter the parameters, in order to know which parameter settings give us the best performance, but we did not make any changes to the algorithm itself. While it is possible to increase the performance when certain changes are made to the algorithm. So an other possibility of future work is to test whether alterations to the TABU algorithm can lead to better performance.

In all the algorithm performance tests in this thesis, we did not calculate the time needed to build the network in the overall performance scale. This is mainly because we do not give much preference to how long the algorithms would take. In the tests we performed the time taken by every algorithm was mostly dependent on how large the dataset is. For datasets with the size of our target data the duration of all algorithms was a matter of seconds. That's why we gave preference to the better performing algorithm, even if the algorithms execution time is higher.

In the tests of section 7 all networks were also created very fast, a matter of seconds. There may be algorithms which search more thoroughly for a good network structure, by which they take much longer. But in our case the longer time needed is not that much of an issue, when the resulting network structure is significantly better.

Besides that, there are other tabu search based algorithms that can be tested, which may result in finding a better variant.

We also explored the use of background knowledge in the process of constructing a network. It was demonstrated in section 5 that the addition of background knowledge to the network constructing algorithm can lead to the construction of a better network. In addition, we used it to construct a better network in section 7.

An other possibility of future work is to examine the exact influence of background knowledge. To be more precise, how much can does the addition of background knowledge increase the performance of the algorithm. Is there a difference in the performance increase between adding one knowledge relation compared to adding none, and adding 10 relations compared to adding 9. Is there a moment when the performance increase stops or is negligible? Such information would yield insights into which and how much knowledge is necessary in order to obtain better network structures.

When we have a better understanding of the above, we may be able to develop a better algorithm. This may lead to the construction of better causal Bayesian networks from which more can be learned about the domain.

Chapter 9

Tables

Variable (Variable name in network)	Value(s)
At diagnosis	
Period at diagnosis (Period)	1996-1999, 2000-2004, 2005-2009
Age at diagnosis (Age)	<40, 40-60, 60-75, >75 years
Karnofsky score (Karnofsky)	<40, 40-60, >60
History of cancer	Yes or No
History of breast cancer (History of BC)	Yes or No
Genetic predisposition (Predisposition)	Yes (i.e. BRCA 1 or 2 or HNPCC) or No
Co-morbidity [a]	0, 1-5, >5
BMI	<20, 20-25, 25-30, >30
Menopausal state	Pre or postmenopausal
Disease related variables	
Tumor grade (Grade)	Grade 1, 2 and 3
Tumor histology (Histology)	Serous, Mucinous, Endometrioid, Clear cell, Brenner, Undifferentiated, Mixed, Adenocarcin oma not otherwise specified (NOS), Other or Unknown
Initial volume of the tumor (Volume)	0-11, 11-18, >18cm
Metastasis	Yes or No
Ascites	Yes or No
Ultrasound score [b] (Ultrasound)	0, 1, >1
CA125 at diagnosis (CA125 diagnosis)	(<35 or ≥35 U/ml)
FIGO stage (Stage)	Stage I,II,III or IV

Table 9.1: List of selected variables of the ovarian cancer dataset

Primary therapy related variables	
Trial participation (Trial)	Yes or No
Hospital of diagnosis (Hospital)	Academic, teaching, general hospital
Primary debulking performed (Primary debulking)	Yes, No because of condition patient, No because of tumor volume or site, No for other reasons
Gyne-oncologist at primary debulking (Gyne-oncologist)	Yes or No
Debulking hospital (Hospital debulking)	Academic, teaching, general hospital, not appplicable
Volume of tumor residuals after primary debulking (Volume primary)	Macroscopic free, < 1 cm, 1-2 cm, >2 cm-≤ 5 cm, >5cm -≤ 10 cm, >10 cm, Not applicable, Unknown
CA125 after primary debulking (CA125 primary)	(<35 or ≥35 U/ml
Secondary therapy related variables	
Interval debulking performed (Interval debulking)	Yes or No
Volume of tumor residuals after Intervaldebulking (Volume interval)	Macroscopic free, < 1 cm, 1-2 cm, >2 cm-≤ 5 cm, >5cm -≤ 10 cm, >10 cm, Not applicable, Unknown
CA125 after intervaldebulking (CA125 interval)	<35 or ≥35 U/ml
Chemotherapy related variables	
Adequate chemotherapy [c]	Adequate, Inadequate, Not applicable
Response on chemotherapy	Complete remission, Partial remission, Stable disease, Progression of disease, Not applicable, Unknown
CA125 after chemotherapy (CA125 chemotherapy)	<35 or ≥35 U/ml

[a] Charlson co-morbidity score [39]

[b] Ultrasound score following the RMI score used by Tingulstad [40]

[c] At least 6 courses of platin-containing chemotherapy

Table 9.2: List of selected variables of the ovarian cancer dataset

Bibliography

[1] R. R. Bouckaert, *Properties of Bayesian Belief Network Learning Algorithms*, Utrecht University Department of Computer Science, 1994

[2] W. Buntine, Theory refinement on Bayesian networks, Uncertainty Proceedings, 1991, pp. 51–60.

[3] G. F. Cooper, E. Herskovits *A Bayesian method for the induction of probabilistic networks from data*, Journal Machine Learning archive Volume 9 Issue 4, Oct. 1992, Pages 309-347

[4] F. V. Jensen, *Introduction to Bayesian Networks*, 1st edition, Springer-Verlag Berlin, Heidelberg 1996, ISBN:0387915028

[5] Heckerman, D., Geiger, D., and Chickering, D. (1995). *Learning Bayesian networks: The combination of knowledge and statistical data.* Machine Learning

[6] J.H. Kim and J. Pearl, A computational model for causal and diagnostic reasoning in inference systems, *Proceedings of the 8th Internation Joint Conference on Artificial Intelligence,* 1983, pp. 190–193.

[7] D. Koller, N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, 2009.

[8] P.J.F. Lucas, L.C. van der Gaag and A. Abu-Hanna (2004). Bayesian networks in biomedicine and health-care. *Artificial Intelligence in Medicine*, 30: 201–214.

[9] J. Pearl, *Bayesian networks: A model of self-activated memory for evidential reasoning*, Cognitive Systems Laboratory Computer Science Department University of California, 1985.

[10] G. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Mateo, California.

[11] Pearl, J. (2000), *Causality: Models, Reasoning, and Inference*, Cambridge, UK: Cambridge University Press.

[12] J. Pearl and T. Verma, *A theory of inferred causation, Cognitive Systems Laboratory*, Computer science department University of California, Los Angeles

[13] D. Margaritis, *Learning Bayesian Network Model Structure from Data*, School of Computer Science Carnegie Mellon University, May 2003

[14] Tsamardinos I, Brown LE, Aliferis CF (2006). *The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm*, Machine Learning, 65(1), 31–78

[15] F. W. Glover (1986), *Future Paths for Integer Programming and Links to Artificial Intelligence* Computers and Operations Research. 13 (5): 533–549.

[16] D. M. Chickering, D. Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. Machine learning 29, 181-212

[17] A comparison of Bayesian network structure learning algorithms on emergency department ambulance diversion data. Faculty of the Graduate School of Vanderbilt University. August, 2009. Nashville, Tennessee

[18] Fu. A comparison of state-of-the-art algorithms for learning Bayesian network structure from continuous data. Faculty of the Graduate School of Vanderbilt University. December, 2005

[19] F. M. Al-Akwaa, M. M. Alkhawlani. Comparison of the Bayesian Network Structure Learning Algorithms. University of Science and Technology University of Science and Technology, Sanaa, Yemen. International Journal of Advanced Research in Computer Science and Software Engineering. Volume 2, Issue 3, March 2012

[20] M. Scutari. Measures of Variability for Graphical Models. Department of Statistical Sciences research doctorate school in statistical sciences cycle XXII. January 31, 2011

[21] D. Heckermann. Bayesian Networks for Data Mining. Journal. Data Mining and Knowledge Discovery archive. Volume 1 Issue 1, 1997. Pages 79-119

[22] Grzegorczyk M. (2010) An Introduction to Gaussian Bayesian Networks. In: Yan Q. (eds) Systems Biology in Drug Discovery and Development. Methods in Molecular Biology (Methods and Protocols), vol 662. Humana Press, Totowa, NJ

[23] Published: 2018-01-03 Author: Frank E Harrell Jr, with contributions from Charles Dupont and many others. https://CRAN.R-project.org/package=Hmisc

[24] Nir Friedman, Dan Geiger, Moises Goldszmidt, Bayesian Network Classifiers. Machine Learning (1997) 29: 131. https://doi.org/10.1023/A:1007465528199

[25] van Altena AM1, Karim-Kos HE, de Vries E, Kruitwagen RF, Massuger LF, Kiemeney LA. Trends in therapy and survival of advanced stage epithelial ovarian cancer patients in the Netherlands. Gynecol Oncol. 2012 Jun;125(3):649-54.

[26] Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22. URL http://www.jstatsoft.org/v35/i03/.

[27] https://www.r-project.org/about.html

[28] $https://www.cijfersoverkanker.nl/selecties/Dataset_2/img5b042a0195de1$