Czech Technical University in Prague
Faculty of Transportation Sciences
Department of Applied Mathematics

Miroslav Vaniš

Optimization of Bayesian networks and their prediction properties

Doctoral thesis

Supervisor: doc. Ing. Ivan Nagy, CSc.
Ph.D. Programme: Engineering Informatics

Prague 2021

Declaration

This doctoral thesis is submitted in partial fulfillment of the requirements for the degree of doctor (Ph.D.). The work submitted in this dissertation is the result of my own investigation, except where otherwise stated.

I declare that I worked out this thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis. Moreover I declare that it has not already been accepted for any degree and is also not being concurrently submitted for any other degree.

Acknowledgment

I would like to thank my supervisor for his help throughout the creation of this doctoral thesis. I also appreciate all the support I received from my family. I would also like to thank my friends, especially Kryštof Urbanec, Ondřej Hába, Tomáš Třasák and Jan Filipčík for many discussions related to the Bayesian networks.

Abstract

Generally, Bayesian networks can be created in two basic ways: by a structural algorithm or by an expert. This work focuses on the interconnection of these two approaches. The main goal is an algorithm for merging two different Bayesian networks into a final one, which will be better for data analysis. The algorithm works on the principle of selecting edges based on the score of nodes in the input networks. The algorithm is described by simple examples and then mathematically, including a description of its evaluation. Finally, the algorithm is then illustrated on the example of real traffic accident data.

Contents

Introduction

Nowadays, there is a huge amount of data in various areas. Companies can use data in several different ways to mainly achieve the highest profit, the maximal market sale. The medical system tries to utilize data to better set the diagnosis and to decrease the number of sick patients etc. There are countless possibilities for data usage. Data can be processed in many ways and it is crucial to determine the at least approximate purpose of data processing to select the area of appropriate methods suitable for the data and the purpose. For example, a restaurant wants to have information about its guests in some period to schedule food order, a hospital would like to have an overview of its hospitalized patients to better schedule shift work, a town hall plans a development strategy in various areas based on measured data, etc.

One of the areas in the latter example can be traffic problems. A traffic department of the town hall in Prague decided to make some arrangements in order to reduce the number of accidents, mainly serious ones. They provided an annual traffic accident data to our faculty. They were looking for someone who analyzes that data to investigate the causes of serious traffic accidents. The author was always interested in the traffic and the data analysis, he is a graduate of the Faculty of Transport, so he decided to take up this challenge.

The task of causes of serious traffic accidents is actually to find the relations between data variables. This is one of the classical tasks in data processing. There are mainly data-mining methods suitable for this type of problem. If we go to the example more deeply, we are actually interested in the changes of caused variables based on input knowledge. In the other words, do serious traffic accidents affect some data variables more than some others? Generally, one of the most suitable methods for investigation of this task is Bayesian networks.

Bayesian networks are probabilistic graphical models that represent relations (more precisely said conditional independencies) between data variables via network structure and conditional probability tables. Their primary purpose is to find possible causes of some measured result. The common approach of this kind of data analysis is to create a Bayesian network using an algorithm based on data and then to calculate Bayesian inference given prior knowledge. It well corresponds to the task mentioned in the previous paragraph. The way of doing these findings heavily depends on a network structure.

Therefore, we will now focus on the possibilities of its creation. It can be created not only by algorithms but also by experts. The network structure created by algorithms will

often not work well without expert knowledge and on the contrary, the expert cannot find relations hidden in data. It follows that networks created in different ways have their strengths and weaknesses. However, the possibilities of combining these approaches are nowadays limited. This is where the idea of this doctoral work came about: is it possible to create better Bayesian network by combining expert and algorithmic approach?

The goal of this thesis is focused on an algorithm that can merge two original Bayesian networks into one merged network. This merged network should be better for data analysis in terms of quality.

Thesis goals

To appropriately introduce the thesis' goals we would like to define some postulates which are already known:

- Bayesian networks are a useful modeling tool for data analysis.
- The data analysis provided by Bayesian networks mainly consists in recalculation of conditional probability mass functions based on prior knowledge.
- The strength of Bayesian networks is also in the graphical representation of a given task.

The main goals of this thesis are to:

- design a merging algorithm that should provide a better network for data analysis than the expert or algorithm ones,
- mathematically describe the merging algorithm,
- show its usage on a real data problem.

Chapter description

The thesis is divided into several sections. Bayesian networks are a product of the graph and the probability theory so we provide basics and notations of graph and probability terms used further in the thesis in terms of Bayesian networks in the Chapter 1.

The theory of Bayesian networks is focused on the Bayesian theorem with its usage in computing inference. Chapter 2 is very significant to understand data analysis using Bayesian networks. It also includes a section about the network structure algorithm in order to show a difference between it and the merging algorithm. The last part of this chapter focuses on the calculation of node scores, which is necessary for both the merging algorithm and its validation.

Chapter 3 then provides the state of the art focused mainly on the network connection issues and further describes the differences between research already done and this work.

The following chapter 4 contains a description of the merging algorithm. Each step of the algorithm is explained and shown in an example.

The chapter 5 aims at the mathematical description of the Merging algorithm with no examples included. It contains sections related to issues that arose during the development of the algorithm and also explains its evaluation.

The last chapter 6 is an example of using this algorithm on real (traffic accident) data. Two networks are created and these are then merged and it is proved that this merged Bayesian network is better.

Chapter 1

Graph and Probability Theory

This work deals with an algorithm for merging Bayesian networks. To understand the general basics of Bayesian networks and their usage, it is necessary to know the basic concepts of graph theory and probability theory. Terms from graph theory are also widely used in order to describe the merging algorithm process. Let us begin with graph theory.

1.1 Graph theory

The basic concepts of graph theory are mainly drawn from [1, p. 2–6].
The basic term of graph theory is the graph itself. It is defined as

- a set of **nodes** (vertices), $V = \{X_1; X_2; \ldots; X_n\}$,
- a set of **edges** (arcs, lines), $E = \{\{X_1, X_2\}, \{X_1, X_3\}, \ldots\}$.

The graph is then written as $G = (V, E)$.
Each edge is defined by a subset from V. This type of graph is called an **undirected graph**. Its example is shown on see Figure 1.1.

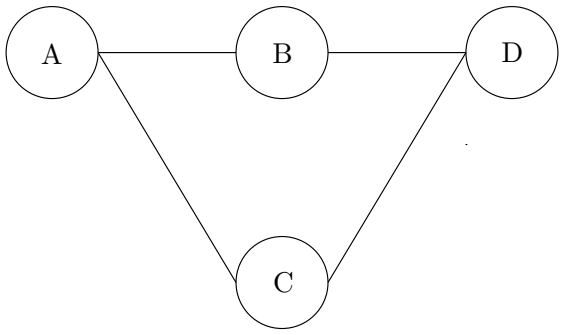

Figure 1.1: The undirected graph with four nodes, $V = \{A, B, C, D\}$, and four edges, $E = \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}\}$

If the two nodes (X_i, X_j) within each edge are ordered, then the edge

$$e = (X_i, X_j) \tag{1.1}$$

is oriented and has a direction from a node X_i to a node X_j but not conversely. It is worth noting here that the oriented edges are marked with parentheses.

If all edges have directions, the graph is called a **digraph** or **directed graph**.

The classical edge notation (equation (1.1)) is without indexes. When describing the merging algorithm, it was found that it is appropriate to introduce another notation in order to simplify the explanation of the algorithm. The edge in equation (1.1) is then rewritten

$$E_{i,j} = (X_i, X_j). \tag{1.2}$$

Further information on the issue of the edge notation is explained in the relevant parts of the algorithm description.

Let us now explain other important terms. A **chain** is a series of nodes where each successive node is connected to the previous one by an edge (regardless of the edge directions). A **path** is a chain with the further constraint that each connecting edge in the chain has a directionality going in the same direction as the chain. A **simple path** is a path that includes unique nodes. A **cycle** is a path that begins and ends at the same node. A **simple cycle** is a cycle where all nodes but start/end nodes are unique. These terms are depicted in Figure 1.2.

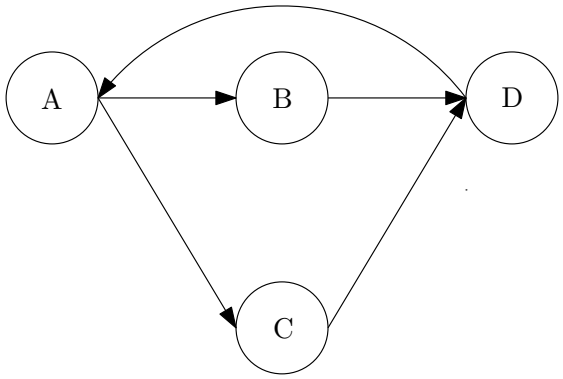

Figure 1.2: The directed graph with four nodes and five edges. The node sequence A-D-B-A is a chain of length 3, but it is not a path. The node sequence A-B-D-A-C is a path of length 4. The node sequence A-B-D is a simple path of length 2. The node sequence A-B-D-A-B-D is a cycle of length 5 and the node sequence A-B-D-A is a simple cycle of length 3.

A **directed acyclic graph (DAG)** is a directed graph that has no cycles. The following terminology is meaningful only for digraphs.

Suppose an oriented edge (X_1, X_2). X_1 is then a **parent** of X_2 and X_2 is a **child** of X_1. Furthermore, there exists an extension of parent/child relationship into ancestor/descendant relationship. If X_1 is then the parent of X_2 and X_2 is the parent of X_3,

then X_1 is an **ancestor** of X_3, and X_3 is a **descendant** of X_1. This relationship can be extended into more than just two sets of parents/child relations. A **family** is a set of nodes in which contains node X and parents of X.

A **forest** is a DAG where each node has either one parent or none. A **tree** is a forest where only one node in a graph has no parent. This node is called the **root**. A **depth** of a node is the number of edges from the node to the tree's root node. A root node will have a depth of 0.

The terms from the previous two paragraphs are shown in the example in Figure 1.3.

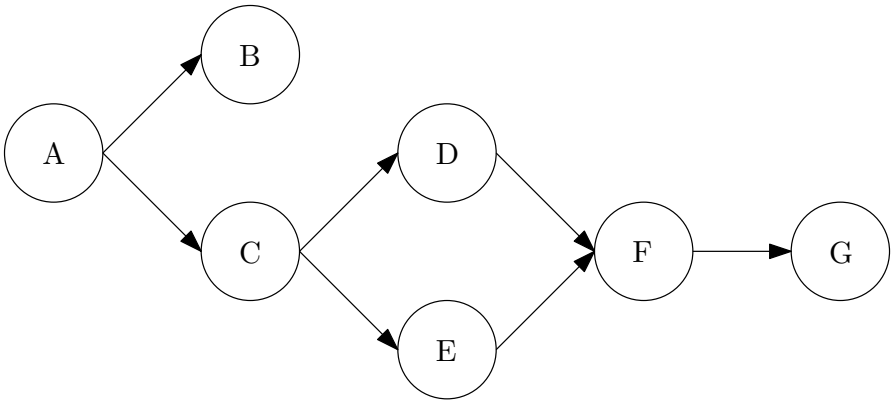

Figure 1.3: This figure shows DAG (directed acyclic graph). Node A is a parent of nodes B and C while nodes B and C are children of node A. Node A is an ancestor of nodes B, C, D, E, F, and G while nodes B, C, D, E, F, and G are descendants of node A. A family of node C is nodes C and A, a family of node F are nodes F, D, and E. This graph is not either a forest or a tree (Node F has two parents).

The terms child and parent are defined between two nodes by a directed edge, the term **adjacent** (or **neighbor**) describes the relationship regardless of the edge direction. The two nodes are said to be adjacent.

The term **complete** describes an undirected graph where every node is connected to all other nodes.

It is clear that graph theory has many other terms. This subsection focused only on terms related to Bayesian networks and the merging algorithm. For more information on graph theory, we recommend the literature [2].

1.2 Probability theory

The next area related to Bayesian networks is probability theory. At the beginning we will give an example on which we will then explain the basic concepts. This example will contain three variables with two possible values and a sample of their data.

We will choose an example from the field of transport, so the variables will be Accident Severity, Speed, and Year of Production. They have the following values:

- Accident Severity (A): light (0), serious (1)
- Year of Production (S): less than 5 years (0), more than 5 years (1)
- Speed (Y): under max. speed limit (0), over max. speed limit (1)

In this chapter we will mostly use abbreviations in parentheses for these variables and their values. The data sample is given in Table 1.1.

Table 1.1: The data sample for the variables Accident Severity (A), Speed (S), and Year of Production (Y)

Record	A	S	Y
1	1	1	1
2	0	1	0
3	1	1	0
4	0	0	0
5	1	0	0
6	0	0	1

Now let us focus on the probability calculation. For example, we may ask what is the probability that $A = 0$,$S = 0$ and $Y = 0$. In this case, the result is simple. Let us look at the table and find the rows that match the query and divide it by the number of data records

$$P(A = 0, S = 0, Y = 0) = \frac{1}{6}. \tag{1.3}$$

The $P(A, S, Y)$ is called the **joint probability mass function** [3, p. 17]. If we generalize the previous calculation, it holds that the joint probability mass function for any combination of variables is computed as a frequency of a defined query divided by the number of records.

With more variables, their values, and more data records, calculating the joint probability mass function becomes more complicated. Fortunately, there are some possibilities to simplify these calculations. In order to use them, we will introduce additional functions.

In the previous example, we asked the probability that three variables take on a value at the same time. If we ask only about the probability of only one value, e.g. $Y = 1$, the notation will be as follows

$$P(Y = 1) = \frac{2}{6}. \tag{1.4}$$

In general, this probability function $P(Y)$ is called a **marginal probability mass function**[3, p. 18].

We may also be interested in the probability of a variable under certain conditions, e.g. $Y = 1$ under conditions $A = 0$

$$P(Y = 1|A = 0) = \frac{1}{6}. \tag{1.5}$$

Generally, this is the case of a **conditional probability mass function**[3, p. 18, 21] $P(Y|A)$. This also means, among other things, that the variables Y and A are interdependent. In other words, the probability that the variable Y acquires a certain value depends on the probability of the values of the variable A and vice versa.

The first simplification of the joint probability mass function calculation consists in the application of the so-called **chain rule**[4, p. 48]. For any two variables A,B that are marginally independent (see below in this section) it applies that

$$P(A, B) = P(A|B)\, P(B). \tag{1.6}$$

We say that the joint probability mass function is **factorized** by the marginal and conditional probability mass function. If we use the variables from the example and extend the chain rule for three variables, we obtain

$$P(A, S, Y) = P(A|S, Y)\, P(S|Y)\, P(Y). \tag{1.7}$$

We can apply the chain rule when calculating the previous example (equation 1.3)

$$\begin{aligned} P(A = 0,&S = 0, Y = 0) = \\ &= P(A = 0|S = 0, Y = 0)\, P(S = 0|Y = 0)\, P(Y = 0) = \\ &= \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{6} = \frac{1}{6}. \end{aligned} \tag{1.8}$$

We can also use the chain rule with a different order of variables

$$\begin{aligned} P(A = 0,&S = 0, Y = 0) = \\ &= P(S = 0|A = 0, Y = 0)\, P(Y = 0|A = 0)\, P(A = 0) = \\ &= \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{6}. \end{aligned} \tag{1.9}$$

When calculating a particular probability, it happens that we do not know the values of the conditional probability function, e.g. $P(S|Y)$, but we know the probability function $P(Y|S)$.

A connection between these conditional probability mass functions is done by **Bayesian rule**[5, p. 130]:

$$P(S|Y) = \frac{P(Y|S)\, P(S)}{P(Y)}. \tag{1.10}$$

If we substitute the probability function from equation (1.10) into equation (1.7), we get

$$P(A, S, Y) = P(S|A, Y)\, \frac{P(A|Y)\, P(Y)}{P(A)}\, P(A) = P(S|A, Y)\, P(A|Y)\, P(Y). \tag{1.11}$$

This result also corresponds to the use of a chain rule in a different order of variables.

Next we will show another possibility how to adjust the joint probability mass function calculation. In explaining the conditional probabilities, we stated that the variables are interdependent (equation (1.5)). However, this may not always be the case. We may know (or assume for some reason) that the variables do not depend on each other. We say that they are **marginally independent**[4, p. 45]. Mathematically speaking,

$$P(Y) = P(Y|A). \tag{1.12}$$

the conditional probability function became the marginal one. The same also applies conversely,

$$P(A) = P(A|Y). \tag{1.13}$$

For the joint probability mass function then applies

$$P(A, Y) = P(A)\, P(Y). \tag{1.14}$$

We will now proceed to the next term. It is a bit more complicated to explain. To avoid confusion with the previous examples, we introduce the new variables B, C and D. These variables will represent events and will have only two values: either an event occurs or it does not.

Assume the following situation: the events B and C are marginally independent. There is also another event D that is related to both the B and C events. This situation is graphically illustrated in Figure 1.4.

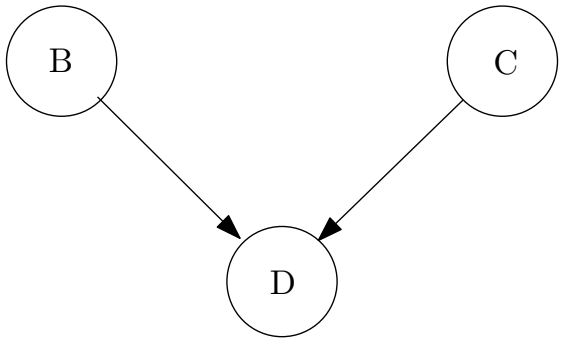

Figure 1.4: Illustrative image in order to explain conditional dependence

It follows that, if the event B and D occur simultaneously the probability of occurrence of the event C will decrease because its relation to the event D is less necessary as an explanation for the occurrence of the event D. Identically, if the events C and D occur simultaneously, the event B occurring will decrease the probability of occurrence of the event D.

We say that the events B and C are **conditionally dependent** [4, p. 46] on each other given the event D. The latter case is mathematically rewritten as inequality

$$P(B|D) \neq P(B|C, D). \tag{1.15}$$

Suppose that these events are some positive things, that we buy something new, such as a car (event B) or a television (event C). Our mood will then correspond to event D and will increase with any purchase. Imagine a situation where we meet a friend and he sees that we are in a good mood (event D is occuring) and that we arrived in a new car (event B is occuring). A friend decides that our mood has improved mainly due to the purchase of a new car (event B) and that my mood could have improved by buying a television (event C) is no longer so important for him. This corresponds mathematically to the equation (1.15).

Finally, we introduce a concept that is closely related to Bayesian networks. In the previous paragraphs, we had such an example that variables are independent of each other and, based on some other variable, become conditionally dependent. Now let us look at the exact opposite. The two variables (events E and F), are interdependent, but given the value of another variable (event G), they become **conditionally independent**.

We will start from the marginal independence (equation (1.14)) and we just relabel the variables from Y,A to E,F and add the conditional variable G. Then it applies

$$P(E, F|G) = P(E|G)\, P(F|G). \tag{1.16}$$

As an example from life, we can imagine that the variable E is a person's height and the variable F is his vocabulary. These are two dependent variables (the higher a person's height, the higher his vocabulary). However, if we introduce the variable G into our example, which will be the age of this person, the variables E and F will become completely independent.

1.3 Chapter summary

This chapter was divided into two basic parts: graph theory and probability theory. In these sections, we have focused on basic concepts that will be useful throughout the work.

If we are to point out the essential parts for the merging algorithm, it is definitely graph theory and its basic concepts such as node, edge, cycle, parent, ancestor, etc.

Regarding the insight Bayesian networks usage as a tool for data analysis, it is necessary to have a good understanding of all the concepts explained in the part of probability theory.

Chapter 2

Bayesian networks

This chapter focuses on Bayesian networks mainly from two perspectives. The first of them is a basic theory and then a demonstration of working with them for data analysis. This is fully sufficient for their use.

The last two subchapters focus on more advanced things. As we know from the introduction, one of the parts of the Bayesian network is its structure. In the penultimate section, we will explain the principle of creating a structure based on data and a demonstrate it on a classical algorithm for these purposes. The last part of this chapter is devoted to the issue of node score, which is necessary for the merging algorithm.

Let us start with the definition of the Bayesian network.

2.1 Definition

A Bayesian network (BN) is described as an acyclic directed graph (DAG) which defines a factorization of a joint probability mass function over the variables that are represented by the nodes of the DAG, where the factorization is given by the directed edges of the DAG.

Mathematically, BNs are models that efficiently encode the joint probability mass function for a large set of variables $\mathbf{X} = \{X_1, , X_n\}$ into DAG. BNs consists of [6, p. 10]:

- Network structure encoding conditional independence assertions about $\mathbf{X}$.
- A set of conditional probability tables (CPTs) which contain local (marginal or conditional) probability mass functions associated with each variable in $\mathbf{X}$.

The nodes in BNs are in one-to-one correspondence with the variables in $\mathbf{X}$ and the edges represent a relation between two particular variables.

Both points of this definition (structure and CPT) can be determined using an algorithm from data or an expert. In the example that will guide us in this chapter, the structure and CPT will be determined by an expert.

2.2 Bayesian network principles

Let us now turn to the demonstration of working with a simple Bayesian network in order to show its basic principles. We will use the example from the previous chapter with variables Year of production (Y), Speed (S), Accident Severity (A). By definition, we introduce a set $\mathbf{X}$ containing these variables (in text, we will use abbreviations again)

$$\mathbf{X} = \{Y, S, A\}.$$

In addition, the Bayesian network operates with dependencies (or independence) assertions between variables (see the first point of the definition). These are usually represented by the structure of the network. For our example, consider the dependencies shown in Figure 2.1.

To fully define the Bayesian network, we should also assign a CPT to each variable. If we look at Figure 2.1, we see that the variable Year of production has no parents. For now, it will be assigned a marginal probability function $P(Y)$. The same is true for the variable Speed, hence $P(S)$. These assignments are possible due to conditional independence, we will return to this in a moment.

The Accident Severity variable has two parents: Speed and Year of Production. Therefore, it will be assigned a conditional probability function $P(A|S, Y)$. Specific probabilities for the values of these variables are not yet needed. We will return to them when calculating various probability functions in the section 2.4.

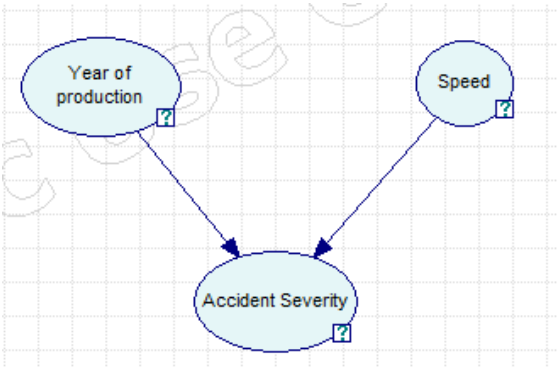

Figure 2.1: A simple Bayesian network.

The Speed and Year of production nodes do not have a direct link to each other. This means that they are marginally independent. If we know the value of the Accident Severity variable with certainty, the Speed and Year of Production variables become conditionally dependent given the variable Accident Severity.

Bayesian networks allow us to calculate different probability mass functions (see Section 1.2) based on knowledge of certain values of variables. In terms of Bayesian networks, this knowledge is called **evidence** [4, p. 9]. The phrase "set an evidence" is also often used, which means that the variable is set with a 100% probability to a certain value.

We know from probability theory that the joint probability mass function of $\mathbf{X}$ can be computed via the chain rule

$$P(A, S, Y) = P(A|S, Y)\, P(S|Y)\, P(Y). \tag{2.1}$$

Due to the fact that the variables S and Y are marginally independent, we can simplify the calculation to

$$P(A, S, Y) = P(A|S, Y)\, P(S)\, P(Y). \tag{2.2}$$

If we generalize this equation for N variables in a set $\mathbf{Y} = \{y_1, y_2, \ldots, y_N\}$, we get a general formula for the joint probability mass function:

$$P(\mathbf{Y}) = \prod_{n=1}^{N} P(y_i|\mathrm{Pa}(y_i)), \tag{2.3}$$

where $\mathrm{Pa}(y_i)$ means parents of a variable y_i.

This general formula is based on one very important premise, without which the Bayesian network could not function. **Each variable is conditionally independent of its non-descendants, given its parents** [7, p. 31]. This means that each variable, or its CPT, is sufficient to consider only the parents of that variable. This has the advantage that the number of parameters of the entire Bayesian network is significantly reduced.

It is also an explanation of why, in the example in Figure 2.1, only marginal probability functions $P(Y)$ and $P(S)$ could be assigned to the variables Y and S. These variables have no parents, so the probability mass functions are simplified to marginal ones.

The relationship between the joint probability mass function and the Bayesian network is not one to one. In the other words, it is not true that one joint probability mass function corresponds to one Bayesian network. Let us take an example.

We have the factorization of the joint probability mass function $P(A, S, Y)$ that corresponds to the structure depicted in Figure 2.1 and equation (2.2).

By applying Bayes' rule to the term $P(A|S, Y)$ in equation (2.2) we get

$$P(A, S, Y) = \frac{P(Y, S|A)\, P(A)}{P(Y)\, P(S)}\, P(S)\, P(Y) \tag{2.4}$$

and assuming conditional independence between the variables Y and S given A, it holds that

$$P(Y, S|A) = P(Y|A)\, P(S|A). \tag{2.5}$$

Substituting this equation into equation (2.4) we get the joint probability mass function

$$P(A, S, Y) = P(Y|A)\, P(S|A)\, P(A). \tag{2.6}$$

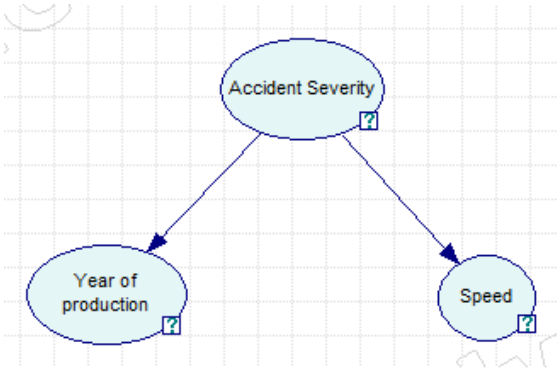

Figure 2.2: A simple Bayesian network with a different causality compared to Figure 2.1.

This factorization of the joint probability mass function corresponds to the Bayesian network shown in Figure 2.2.

It is therefore clear that the difference between these networks (Figure 2.1 and 2.2) is a consequence of the conditional independence assertions. These are reflected in the orientation of the edges (dependencies) in the structures.

We have shown that the Bayesian network corresponds to one particular factorization of the joint probability mass function. We should take this fact into account when solving tasks using Bayesian networks.

2.3 Software used for Bayesian networks

Already in the previous section we could see two figures of the Bayesian network structure. They were created using GeNie software [8] which will be used in the work not only as a graphical tool to display our results but also as a software used to model Bayesian networks.

Of course, there are other tools for working with Bayesian networks. GeNie was chosen for several reasons:

- It is free for scientific purposes.
- It can do all the basic activities associated with the Bayesian network (drawing the structure of Bayesian networks, doing Bayesian inference, working with files etc.)
- It contains algorithms for creating the structure of Bayesian networks from data.
- It contains algorithms for calculating CPT based on a given structure from data.
- It also contains the Smile library [9], which can be used to work with Bayesian networks in the programming languages Java, C# etc.

In the next section, we will use GeNie to verify our calculations in Bayesian inference examples.

2.4 Bayesian inference

In terms of Bayesian networks, Bayesian inference is generally the calculation of relevant probability mass functions based on new incoming evidence based on the defined CPTs.

In order to show this inference by example, it is necessary to determine individual CPTs (probability mass functions). We will continue with the same example from the previous section, for clarity the structure of the network is once again shown in Figure 2.3.

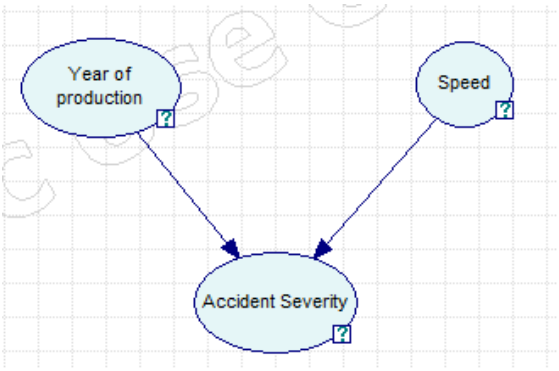

Figure 2.3: A simple Bayesian network.

In the previous section, we explained the assignment of the CPT to individual variables, so it holds that the CPT of the variable Year of Production (Y) corresponds to the probability mass function $P(Y)$, where the probabilities for specific values are given in Table 2.1.

Table 2.1: $P(Y)$ - Year of Production

Year of Production	Y	$P(Y)$
more than 5 years	0	0.7
less than 5 years	1	0.3

The same is true for the Speed variable, whose CPT is in Table 2.2. The last variable in the network is Accident Severity. Based on the structure, it depends on the two previous variables, and therefore it will be assigned a CPT that corresponds to the conditional probability function $P(A|Y,S)$. This CPT is in Table 2.3.

Just as a reminder, the CPTs and structure is determined by an expert.

Finally, we have the Bayesian network that fully meets the definition in Section 2.1.

Table 2.2: $P(S)$ - Speed

Speed	S	$P(S)$
over the max. speed limit	0	0.1
under the max. speed limit	1	0.9

Table 2.3: $P(A|Y,S)$ – Accident Severity

Year of Production		0		1	
Speed		0	1	0	1
Accident severity	serious = 0	0.7	0.35	0.4	0.1
	light = 1	0.3	0.65	0.6	0.9

Now let us move on to Bayesian inference. We start by saying that we may be interested in the marginal probability functions of variables that do not have them assigned as CPTs.

2.4.1 Marginal probability mass function

In our example, there is only one variable, Accident Severity, that does not have a CPT which corresponds to the marginal probability mass function. What we know is the CPT, which depends on the parents of this variable, corresponding to $P(A|S,Y)$. The calculation of the marginal probability function is performed using the so-called **marginalization** [4, p. 32]. This means recalculation over all variables contained in the condition

$$P(A) = \sum_{Y,S} P(A|Y,S)\, P(Y,S). \tag{2.7}$$

We also know that the variables Y and S are marginally independent and therefore it holds that

$$P(Y,S) = P(Y)\, P(S) \tag{2.8}$$

For completeness, the multiplication in the previous equation is called the tensor product.

Let us show the calculation of $P(A)$ with specific values. We begin by calculating the joint probability mass function $P(Y,S)$. In the case of two variables, this can be expressed in a table. This will contain probabilities for all combinations of values of the variables Y and S. The relevant values of probabilities will be taken from Tables 2.2 and 2.1.

The result is then in Table 2.4.

Each cell corresponds to a combination of variables S and Y, and at the same time this combination occurs in CPT (Table 2.3) of the variable A. E.g. the second column in the CPT of the variable A corresponds to a combination of the variables Y = 0 and

Table 2.4: Joint probability mass function $P(Y,S) = P(Y)\,P(S)$. The probabilities are taken from Tables 2.2 and 2.1.

Year of Production / Speed	0	1
0	$0.7 \cdot 0.1 = 0.07$	$0.3 \cdot 0.1 = 0.03$
1	$0.7 \cdot 0.9 = 0.63$	$0.3 \cdot 0.9 = 0.27$

S = 1. Therefore, if we are interested for A = 0, according to equation (2.7) it's a fact, that

$$\begin{aligned} P(A=0) = & P(A=0|Y=0,S=0)\,P(Y=0,S=0) + \\ & P(A=0|Y=1,S=0)\,P(Y=1,S=0) + \\ & P(A=0|Y=0,S=1)\,P(Y=0,S=1) + \\ & P(A=0|Y=1,S=1)\,P(Y=1,S=1). \end{aligned} \tag{2.9}$$

Now it is enough to substitute specific values from the tables $P(Y,S)$ and $P(A|S,Y)$. The numerical results are shown in Table 2.5.

Table 2.5: Marginal probability mass function $P(A)$, the input values taken from Tables 2.3 and 2.4

Accident Severity	Probability
serious $(A=0)$	$0.7 \cdot 0.07 + 0.35 \cdot 0.63 + 0.4 \cdot 0.03 + 0.1 \cdot 0.27 = 0.3085 = 30.85\%$
light $(A=1)$	$0.3 \cdot 0.07 + 0.65 \cdot 0.63 + 0.6 \cdot 0.03 + 0.9 \cdot 0.27 = 0.6915 = 69.15\%$

In Figure 2.4, we see that GeNie also calculates these marginal probabilities without any problems.

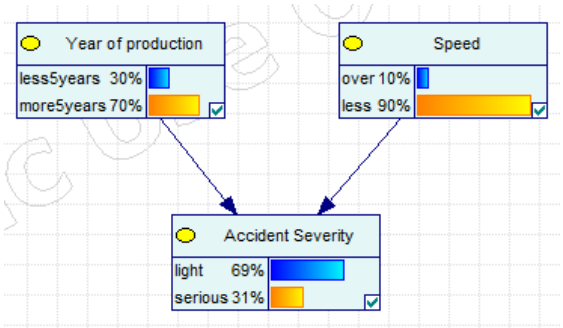

Figure 2.4: Marginal probability mass functions graphically depicted in GeNie.

We have now shown the calculation of the marginal probability function on a simple

example. Of course, the principle of calculation will always be the same even when the variable has more parents and the variables have more values.

We will now move on to the calculation of probabilistic functions, assuming evidence.

2.4.2 Conditional probability mass function

This section is called conditional probability mass function due to the fact that when setting evidence, we will always be interested in a conditional probability mass function given the evidence [10].

We distinguish two basic procedures based on the evidence location:

- causal (top-down) inference,
- diagnostic (bottom-up) inference.

If we are interested in a conditional probability mass function based on evidence placed in one of the ancestors, we call this procedure causal (top-down) inference. On the other hand, if we set the evidence into any of the descendants it is a diagnostic inference.

Let us start with the simpler of these, with causal inference.

Causal inference

We will continue with the example we use in this chapter. For example, we might ask what the conditional probability mass function of the Accident Severity variable will be if we focus only on cases of accidents where the maximum speed has been exceeded, i.e. $S = 0$. Mathematically written $P(A|S = 0)$.

This query meets the Causal inference condition, the evidence is set to the parent of the queried variable.

When calculating any conditional probability mass function in Bayesian network, we must always use what we have available. In our example, we know these probability mass functions $P(Y)$, $P(S)$ and $P(A|S, Y)$.

Therefore, if we calculate $P(A|S = 0)$, we must again use marginalization and also the chain rule

$$P(A|S = 0) = \sum_{Y} P(A|S = 0, Y)\, P(Y) \tag{2.10}$$

to get the equation in which we already know all its members.

If, for example, we are interested in the value for $A = 0$ then

$$\begin{aligned} P(A = 0|S = 0) =& P(A = 0|S = 0, Y = 0)\, P(Y = 0) + \\ & P(A = 0|S = 0, Y = 1)\, P(Y = 1). \end{aligned} \tag{2.11}$$

Specific values of probabilities can be found in Table 2.1 for the variable Y and in Table 2.3 for the variable A. The numerical results are computed in Table 2.6.

Table 2.6: Conditional probability mass function $P(\mathrm{A}|\mathrm{S}=0)$, the input values taken from Tables 2.1 and 2.3

Accident Severity	Probability
serious $P(A=0\|S=0)$	$0.7 \cdot 0.7 + 0.4 \cdot 0.3 = 0.61 = 61\%$
light $P(A=1\|S=0)$	$0.3 \cdot 0.7 + 0.6 \cdot 0.3 = 0.39 = 39\%$

$$P(A|S=0) = \sum_{Y} P(A|S=0,Y)\, P(Y). \tag{2.12}$$

The marginal probability mass function $P(A=0) = 31\%$ and the conditional probability mass function given the evidence $P(A=0|S=0) = 61\%$ differs for about 30%. The exploration of differences between marginal and conditional probability mass functions is an example of typical Bayesian network usage in terms of data analysis.

The example of the calculation of $P(A|S=0)$ in GeNie is shown in Figure 2.5.

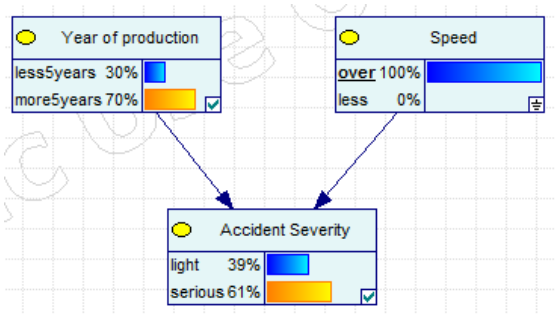

Figure 2.5: Conditional probability mass function $P(A|S=0)$ graphically depicted in GeNie.

Diagnostic inference

In the last section, we set evidence into a parent node and examined its effect on his child. Now we will show the opposite. We will set evidence into the child node and examine its effect on its parent nodes.

Again, we will stick to our familiar example and set evidence in the variable Accident severity to serious, i.e. $A=0$ and we will be interested in its impact on the conditional probability mass function of Speed variable $P(S|A=0)$.

We must again start from the probability mass functions by which the Bayesian network is defined, i.e. $P(Y)$, $P(S)$ and $P(A|S,Y)$. In order to compute $P(S|A=0)$, we will be forced to use Bayes' rule here

$$P(\mathrm{S}|\mathrm{A}=0) = \frac{P(\mathrm{A}=0|\mathrm{S}) \cdot P(\mathrm{S})}{P(\mathrm{A}=0)}. \tag{2.13}$$

The probability $P(A = 0)$ is equal to one due to the evidence setting and $P(S)$ is already known.

The probability mass function $P(A = 0|S)$ is very similar to the function from the previous section. As there, we use marginalization and the chain rule to calculate it

$$P(A = 0|S) = \eta \sum_{Y} P(A = 0|S, Y)\, P(Y). \tag{2.14}$$

We introduce the auxiliary variable η (also called normalization coefficient) [4, p. 34] here, because using Bayes' rule and marginalization, the condition of the probability function that the sum of all probabilities is equal to one would stop to apply.

We now substitute equation (2.14) into equation (2.13) and solve it for $S = 0$

$$\begin{aligned} &P(S = 0|A = 0) = \eta \sum_{Y} P(A = 0|S = 0, Y)\, P(Y)\, P(S = 0) = \\ &= \eta(P(A = 0|S = 0, Y = 0)\, P(Y = 0)\, P(S = 0) + \\ &+ P(A = 0|S = 0, Y = 1)\, P(Y = 1)\, P(S = 0)) \end{aligned} \tag{2.15}$$

All probabilities in this equation are known, again we get them from the corresponding tables (2.1, 2.2 and 2.3). The numerical results are given in Table 2.7.

Table 2.7: Numerical results of $\eta P(S|A = 0)$, the input values taken from Tables 2.1, 2.2 and 2.3

Speed	$\eta P(S\|A = 0)$
over the max. speed limit, S = 0	$0.7 \cdot 0.7 \cdot 0.1 + 0.4 \cdot 0.3 \cdot 0.1 = 0.061$
under the max. speed limit, S = 1	$0.35 \cdot 0.7 \cdot 0.9 + 0.1 \cdot 0.3 \cdot 0.9 = 0.2475$

We know that the following must apply

$$P(S|A = 0) = 1. \tag{2.16}$$

and therefore it holds true that

$$\eta = \frac{1}{\sum_S P(S|A = 0)} = \frac{1}{0.061 + 0.2475} \doteq 3.2415. \tag{2.17}$$

Then it is enough to multiply the probabilities from Table 2.7 by this coefficient and we have the following results:

$$\begin{aligned} &P(S = 0|A = 0) = 0.061\, \eta = 0.1977 \doteq 20\% \\ &P(S = 1|A = 0) = 0.02475\, \eta = 0.802 \doteq 80\%. \end{aligned} \tag{2.18}$$

The example of the calculation of $P(S|A = 0)$ in GeNie is shown in Figure 2.6.

And now again we can examine the differences between the marginal and conditional probability functions.

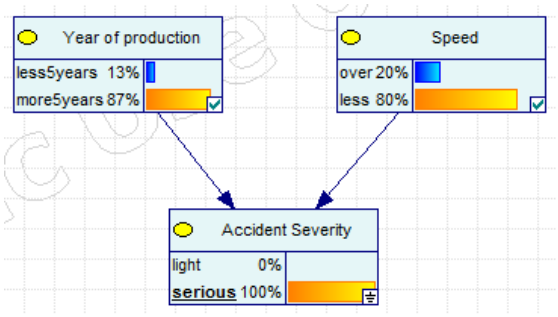

Figure 2.6: Conditional probability mass function $P(S|A = 0)$ graphically depicted in GeNie.

2.4.3 Inference conclusion

In this section on Bayesian networks, we have shown its basic possibilities of data analysis. For more complex Bayesian networks, the so-called inference algorithms are used, but they are also based on these procedures.

A typical example of data analysis may be a comparison of a marginal and conditional probability function or the influence of not only parents or children on a particular node, but also on a node at the same depth, ancestors and descendants.

2.5 Structure algorithms

Until now, we have thought that the structure of the Bayesian network is determined by an expert. This section deals with the design of a structure using a data-based algorithm. First we will talk about these algorithms a little more generally and then we will discuss one algorithm in more detail and show an example.

2.5.1 Algorithm principles

Algorithms for structure design are divided into two basic types

- score-based algorithms [11, p. 18],
- constraint-based algorithms [11, p. 21].

The first type of algorithm, score-based, works with an idea of a goal function that has to be maximized. This leads to problems related to a high number of possible structures. Algorithms typically do not go through all the combinations of variables and their relations (brute force approach). The heuristic methods which reduce the number of possible structures to an acceptable amount are used instead. Also, some prior knowledge at the beginning of searching the structure is helpful. It can decrease the solution time and the number of possible structures.

On the other hand, the constraint-based algorithms are based on finding the relations between input variables. This actually means that they are based on statistical independence tests, so the following assumptions must be met in order to use the algorithms

- causal sufficiency – we have measured all the values of the measurable variables,
- Markov assumption – any variable is independent of all its nondescendants in BN structure, given its parents.

One of the representatives of the constraint-based methods is the Sparse Graph Search (SGS) algorithm. The disadvantage of this approach lies in its weak robustness. It means that the significant changes in the final structure can be caused only by a very small change at the beginning.

Let us now look at an example of a score-based algorithm.

2.5.2 Bayesian search algorithm

The representative of the score-based algorithm is the Bayesian search [12]. It was one of the first algorithms that allow searching a structure with a depth greater than one. There are three assumptions to be fulfilled before running the algorithm:

- the variables are discrete,
- records occur independently,
- there are no missing values.

For a family of score-based algorithms, it always has its goal function assigned to it. The same is true for Bayesian search [12, p. 321]

$$f(BS) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \tag{2.19}$$

This function is complicated at first glance. Let us now explain the individual variables:

- BS specifies the structure for which this function is calculated,
- n is the number of variables in a data sample,
- q_i is the number of possible values of parents of variable x_i,
- r_i is the number of possible values of variable x_i,
- N_{ijk} is the number of records in which variable x_i has the value v_{ik} and the parent (or the combination of the parents) has the value w_{ij},
- N_{ij} is the sum N_{ijk} over $k \rightarrow N_{ijk} = \sum_{k=1}^{r_i} N_{ijk}$.

In the next subsection, we will show the calculation of the specific value of the goal function. This should also lead to a better understanding of the previous formula.

The value of this goal function is calculated for each proposed structure. Then the values are compared and the highest value of them corresponds to the best structure.

Example

We will use the already known example of transport with three variables (A, S, Y) and their two values (0,1).

We said that the goal function is always calculated for a given structure. Let us now calculate the value for the known structure in Figure 2.7.

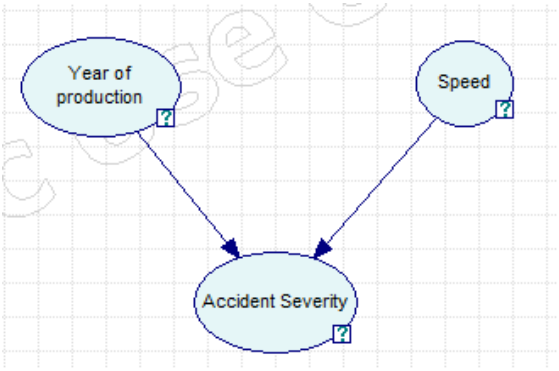

Figure 2.7: A sample structure of the Bayesian network

The data we will use for this calculation are given in Table 2.8. At the moment we will not deal with their origin, they can be real or simulated.

Table 2.8: Data for determing structure using Bayesian search algorithm

Record	Year of production (Y)	Speed (S)	Accident Severity (A)
1	1	1	0
2	0	1	0
3	0	1	0
4	1	1	0
5	1	0	1
6	1	1	1
7	0	0	1
8	1	1	0
9	1	1	0
10	1	1	1

Let us start with the variable Y. Hence, for the following calculation, $i = 1$. There is no consideration about index j because Y has no parents. Therefore,

$$\frac{(r_1 - 1)!}{(N_1 + r_1 - 1)!} N_{10}!\, N_{11}! = \frac{(2-1)!}{(10+2-1)!}\, 3!\, 7!,$$

where $r_1 = 2$ is the number of the variable's Y values (0 or 1), $N_{10} = 3$ is the number of

Y = 0, $N_{11} = 7$ is the number of $Y = 1$ and $N_1 = 10$ is their sum. All values are taken from Table 2.8.

The calculation of the variable S is performed in the same way as for variable Y. Only the variable $i = 2$,

$$\frac{(r_2 - 1)!}{(N_2 + r_2 - 1)!} N_{20}! \, N_{21}! = \frac{(2-1)!}{(10+2-1)!} 2! \, 8!.$$

The last variable differs from the previous two in that it has two parents. If we look at the explanation of the variable N_{ijk}, we find that we are interested in the values of both parents (Y, S) for the value of A.

For this reason, we will create an auxiliary variable. Each of its values will then represent a combination of the values of the parents. The specific values of the assigned combinations are listed in Table 2.9.

Table 2.9: The values of converted variable SY

Year of production	Speed	SY
0	0	0
0	1	1
1	0	2
1	1	3

For each value of the variable SY, we then calculate the appropriate number for N_{ijk}.

- $SY = 0 \rightarrow N_{200} = 0$
- $SY = 1 \rightarrow N_{201} = 2$
- $SY = 2 \rightarrow N_{202} = 0$
- $SY = 3 \rightarrow N_{203} = 4$

As an example, the combination of $Y = 0$ and $S = 1$ occurs twice in Table 2.8, it corresponds to $SY = 1$ and therefore it holds that $N_{201} = 2$.

Their sum is then $N_{20} = 6$.

Again, we increase the value of the variable, $i = 3$. The part of the goal function corresponding to $A = 0$ is

$$\frac{(r_3 - 1)! \, N_{300}! \, N_{301}! \, N_{302}! \, N_{303}!}{(N_{30} + r_1 - 1)!} = \frac{(2-1)! \, 0! \, 2! \, 0! \, 4!}{(6+2-1)!}.$$

The same must then be done for $A = 1$

$$\frac{(r_3 - 1)! \, N_{310}! \, N_{311}! \, N_{312}! \, N_{313}!}{(N_{31} + r_1 - 1)!} = \frac{(2-1)! \, 1! \, 0! \, 1! \, 2!}{(4+2-1)!}.$$

The resulting goal function will then be the product of all calculated members

$$f(1) = \frac{(2-1)!\,3!\,7!}{(10+2-1)!}\frac{(2-1)!\,2!\,8!}{(10+2-1)!}\frac{(2-1)!\,0!\,2!\,0!\,4!}{(6+2-1)!}\frac{(2-1)!\,1!\,0!\,1!\,2!}{(4+2-1)!} = \\ = 2.43 \times 10^{-10}. \tag{2.20}$$

This value alone does not say much. Therefore, we calculated the goal function for the structure shown in Figure 2.8 on the same data (in Table 2.8)

$$f(2) = 2.73 \times 10^{-9}. \tag{2.21}$$

Based on the comparison of these results, we can say that according to the Bayesian search algorithm, the second network structure (Figure 2.8) fits the data more than the first network structure (Figure 2.7).

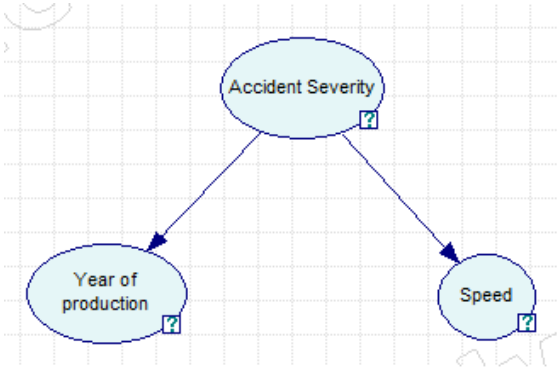

Figure 2.8: A different Bayesian network structure.

2.5.3 Structure algorithms conclusion

In this section, we have provided general information about structural algorithms and shown a simple example in order to demonstrate the basic principles of them. During the study of Bayesian networks, this area was important for understanding how such a Bayesian network originates. At the same time, the idea was born to create such an algorithm that will allow the connection of two networks (regardless of how these networks were created).

2.6 Score of node

This section is a bit different from the others, because it is not necessary to calculate the node score at all in terms of Bayesian network data analysis. The main reason why we will talk about this issue here is that it is one of the inputs to the merging algorithm. In addition, this score plays an important role in the validation of the Merging algorithm.

In this section, we will describe two options for counting node scores. The method of calculating these scores is not essential for the merging algorithm, so it is possible to use some other variant of node evaluation.

First, it is necessary to introduce some concepts in the field of modeling. Each variable in the Bayesian network can be viewed as a discrete model that is generally defined [13, p. 6]

$$P(y_t|\psi_t, \Theta) = \Theta_{y_t|\psi_t}, \tag{2.22}$$

where y_t is a model output, ψ_t is a vector containing values of quantities that affect the output at time t, Θ are model parameters and $\Theta_{y_t|\psi_t}$ is the parameter value for the combination of y_t and ψ_t. The first simplification of this model is in our case that the outputs and inputs are not time dependent and therefore we can omit the index t.

Often this model is represented by a table, where the columns specify the regression vector ψ and the rows the output variable y. In Bayesian networks, conditional probability tables (CPTs) correspond exactly to this.

We will show an example of the variable A (from our example of three variables). The CPT of the variable A is in Table 2.10.

Table 2.10: $P(A|Y, S)$ – Accident Severity

Year of Production	0		1	
Speed	0	1	0	1
$A = 0$	0.7	0.35	0.4	0.1
$A = 1$	0.3	0.65	0.6	0.9

The ψ vector in this case contains the Speed and Year of Production variables. We see that the columns correspond to a value combination of variables in the vector ψ and also determine the output y of the model (variable A). In addition, the probabilities listed in the table correspond to the parameters Θ of the model. For example, the specific parameter value for the combination $A = 0$, $Y = 0$ and $S = 0$ can be written as

$$\Theta_{A=0|S=0,Y=0} = 0.7. \tag{2.23}$$

We have not yet considered the origin of the values in the CPT. For the moment, it is enough for us that these values are estimated in some way (by an expert or an algorithm from the data). But what is important to us is the fact that these values are fixed during the score calculation process. And because the values in the CPT are also the values of the model parameters and are estimated, we will denoted them $\hat{\Theta}$.

We said that the variables in our model do not depend on time. However, they will always depend on the input data. So now let us take a little detour and talk about different types of data in terms of modeling.

2.6.1 Data

As we talk about models, we should also talk about their validation. We have already had some data (Table 1.1 or 2.8) in this work several times and we did not solve what

they are at all. In modeling, it is generally true that we divide the data into two basic samples: **training** and **validation data**.

Training data is used to learn the model and also to fit the parameters of the model. In contrast, validation data can be used to tune the model and also to verify it. So far, the data type has not been important, from now on the data type will play a role. Training data will always be used for the node score but we will no longer modify the model according to them.

We can now begin the description of the first method.

2.6.2 Likelihood-based method

The first method is based on the principles of calculating likelihoods. Likelihood is a function generally used as a measure of model quality based on data with unknown parameters. Next, this function, respectively its maximum is often calculated in order to find the parameters that will best fit a model for which is calculated.

However, this is not our case, we know the parameters (CPTs) exactly and we only calculate the likelihood value for these parameters based on the training data.

Likelihood (in the field of discrete models) [13, p. 12] is generally defined as the product of models

$$L_N(\Theta) = \prod_{n=1}^{N} P(y_n|\psi_n, \Theta) \tag{2.24}$$

where N are the number of data records[1].

Because we work a specific model, the Bayesian networks, we know that the output variable always depends only on its parent variables. Therefore, the vector ψ will contain only these variables. Hence

$$L_N(\Theta) = \prod_{n=1}^{N} P(y_n|\mathrm{Pa}(y), \Theta). \tag{2.25}$$

We have said that parameters are equal to our estimated parameters that remain fixed. Therefore,

$$L_N(\Theta = \hat{\Theta}) = \prod_{n=1}^{N} P(y_n|\mathrm{Pa}(y)) \tag{2.26}$$

where estimated parameters $\hat{\Theta}$ act implicitly as known numbers.

Once you assign likelihood parameters, we calculate the one particular likelihood value for those given parameters.

[1]The data record means the values of all variables, not just the value of the calculated variable.

From equation 2.22. then we know that the probability function is directly equal to the estimated parameter. Hence, it applies

$$L_N(\hat{\Theta}) = \prod_{n=1}^{N} \hat{\Theta}_{y_n|\mathrm{Pa}(y)}. \tag{2.27}$$

It is obvious that the parameters are always values between 0 and 1. Because they multiply each other, the result of the likelihood could be very low.

Therefore, the so-called **log-likelihoods** are used. These are characterized by the use of the product rule on the logarithm and thus the product goes to the sum

$$\ln\left(L_N(\hat{\Theta})\right) = \ln\left(\prod_{n=1}^{N} \hat{\Theta}_{y_n|\mathrm{Pa}(y)}\right) = \sum_{n=1}^{N} \ln\left(\hat{\Theta}_{y_n|\mathrm{Pa}(y)}\right). \tag{2.28}$$

Only Equation (2.28) will be used to calculate the score using the likelihood. It is true that **the lower the value of the log-likelihood, the better the data fits on the given Bayesian network**.

Let us show the calculation of such a score using this method on an example.

Example

This example will use the training data listed in Table 2.11 and the Bayesian network structure shown in Figure 2.9.

Table 2.11: Training data

Record no.(n):	A	S	Y
1	1	0	1
2	1	1	0
3	1	0	0
4	0	1	0
5	0	1	1

As an example, let us calculate the log-likelihood for the Year of Production (Y) node. Its CPT is listed in Table 2.12.

Table 2.12: $\hat{\Theta}_Y$ - Year of Production

Year of Production	Y	$\hat{\Theta}_Y$
more than 5 years	0	0.7
less than 5 years	1	0.3

We calculate the log-likelihood based on equation (2.28).

$$\ln\left(L_N(\hat{\Theta})\right) = \sum_{n=1}^{N} \ln\left(\hat{\Theta}_{y_n|\mathrm{Pa}(y)}\right) \tag{2.29}$$

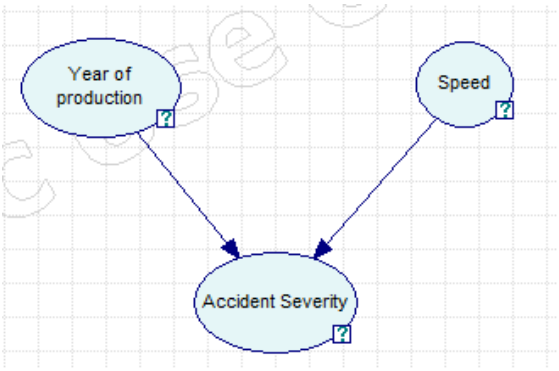

Figure 2.9: A simple Bayesian network.

In this case, the variable y will correspond to the variable Year of Production and the variable has no parents, so the equation is simplified to

$$\ln\left(L_N(\hat{\Theta})\right) = \sum_{n=1}^{N} \ln\left(\hat{\Theta}_{y_n}\right). \tag{2.30}$$

For each value of Y in the data table (Table 2.11) we find the corresponding probability in the CPT (2.12) and substitute

$$\ln\left(L_N(\hat{\Theta})\right) = \ln(0.3) + \ln(0.7) + \ln(0.7) + \ln(0.7) + \ln(0.3) = -3.478. \tag{2.31}$$

This result (-3.478) also serves as the input score of the Y node for the Bayesian network with the structure in Figure 2.9.

Let us move on to another type of node score calculation, prediction error.

2.6.3 Prediction error

First, we must define the term **prediction**[2] [13, p. 29] in terms of Bayesian networks. It is an estimation of an output variable y based on given data.

The estimated value of $\hat{y}$ may, for example, be the maximum of the estimated parameters

$$\hat{y}_{\mathrm{Pa}(y)} = \operatorname{argmax}\left(\hat{\Theta}_{y|\mathrm{Pa}(y)}\right). \tag{2.32}$$

We present the lower index in the estimation of the output variable $\hat{y}$ because the maximum depends on the values of the parents.

For our purposes, it will be more advantageous to predict the mean value of the variable y

$$\hat{y}_{\mathrm{Pa}(y)} = \sum_{i=1}^{k} y_i \, \hat{\Theta}_{y_i|\mathrm{Pa}(y)} \tag{2.33}$$

[2] In some literature, the prediction can be also called classification.

where k denotes the number of values of the variable y and y_i is the particular value of the y.

The prediction error is then the difference between the real value (value obtained from data) of the variable y and its predicted value $\hat{y}$

$$e_n = |y_n - \hat{y}_{\mathrm{Pa}(y)}| \tag{2.34}$$

for one (n-th) data record.

The prediction error for the output variable y is then computed as the sum of the partial prediction errors

$$e_y = \sum_{n=1}^{N} e_n = \sum_{n=1}^{N} |y_n - \hat{y}_{\mathrm{Pa}(y)}|. \tag{2.35}$$

where N are the number of data records.

Example

We will work with the same example (the structure depicted in Figure 2.9 and the training data in Table 2.11) introduced in likelihood section. For example, we will choose a different output variable than last time, node A – Accident severity corresponds to y. Its CPT is in Table 2.13.

Table 2.13: $\hat{\Theta}_{A|Y,S}$ – Accident Severity

Year of Production	0		1	
Speed	0	1	0	1
serious = 0	0.7	0.35	0.4	0.1
light = 1	0.3	0.65	0.6	0.9

First we calculate the mean value (based on equation (2.33) and the values in CPT 2.13) for each value combination of the parents of the output variable y

$$\begin{aligned}
\hat{y}_{Y=0,S=0} &= 0 \times 0.7 + 1 \times 0.3 = 0.3, \\
\hat{y}_{Y=0,S=1} &= 0 \times 0.35 + 1 \times 0.65 = 0.65, \\
\hat{y}_{Y=1,S=0} &= 0 \times 0.4 + 1 \times 0.6 = 0.6, \\
\hat{y}_{Y=1,S=1} &= 0 \times 0.1 + 1 \times 0.9 = 0.9.
\end{aligned} \tag{2.36}$$

For clarity, we will present the training data again. They are listed in Table 2.14.

The calculation of the prediction error based on the training data is following:

$$e_A = |1 - 0.6| + |1 - 0.65| + |1 - 0.3| + |0 - 0.65| + |0 - 0.9| = 3 \tag{2.37}$$

We always take the value of the output variable y (in this case A) from Table 2.14 and subtract it based on the parental values from the corresponding calculated mean in the same data record. In the first row in Table 2.14.

Table 2.14: Training data

Record no.(n):	A	S	Y
1	1	0	1
2	1	1	0
3	1	0	0
4	0	1	0
5	0	1	1

2.7 Chapter summary

This chapter includes general information about Bayesian networks. It begins with the definition of Bayesian networks and continues with the joint probability mass function and its relation to the Bayesian networks. It is explained that Bayesian networks can be described by one joint probability mass function. On the other hand, joint probability mass function can be described by several Bayesian networks based on relations between variables in the mass function.

The chapter continues with the Bayesian inference computation. The computation of marginal and conditional probability mass function is explained in an example.

The next section contains the explanation of algorithms used for structure creation. The process of creating a network structure is then described by one of the structure algorithms.

The Merging Algorithm has one of the inputs the score of each node in Bayesian network. Hence, the last section explains the possibilities of a node score calculation.

All these sections are necessary to get an overview of the Bayesian networks. We would not be able to create the Merging algorithm if we did not know almost this theory. There are large books about Bayesian network theory, for those who are interested we recommend [7].

Chapter 3

State of the art

This chapter deals with the publishing activities in the field of Bayesian networks so far, focusing on an essential area for us, merging Bayesian networks.

We are interested in this area mainly because we are designing our own algorithm for merging networks. We will explain some differences between previously published articles on merging Bayesian networks and our algorithm.

In general, the searched articles on merging Bayesian networks [14, 15, 16, 17] try to meet three basic requirements for a merged Bayesian network:

- preserving the original conditional independencies in original (input) Bayesian networks,
- preserving the characteristics of individual BN parameters (more simply preserving CPTs in individual Bayesian networks),
- avoid any generated cycles.

The last requirement is absolutely necessary, because without it the resulting network will not be Bayesian (this follows from the definition).

The first article [14] deals with a mathematical description of network interconnection structurally using intersection and union operators. Consider two original Bayesian networks depicted in Figures 3.1 and 3.2.

If we apply an intersection to these networks, we get the network in Figure 3.3. At first glance, it can be seen that such a network is then essentially unusable for data analysis.

The second case is the union, which is shown in Figure 3.4. This is the complete opposite, where all links from the original networks are inserted into the merged network.

In addition, it can be seen that although the above requirements are mentioned in this article, none of these operators deals with them.

The last requirement is met, the merged network does not contain cycles. This is achieved thanks to the so-called ancestral ordering, where nodes of the merged network are selected based on the distance from the root node.

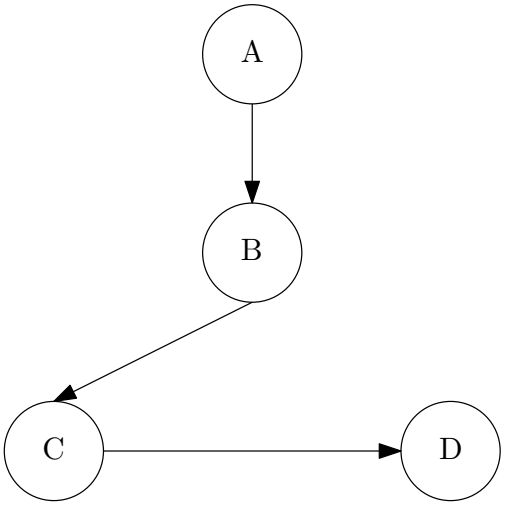

Figure 3.1: The first original Bayesian network

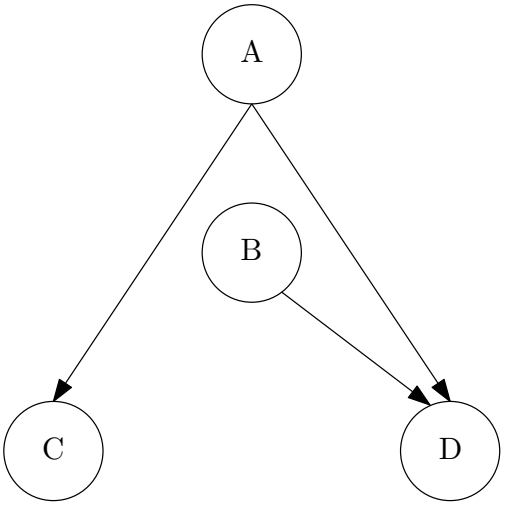

Figure 3.2: The second original Bayesian network

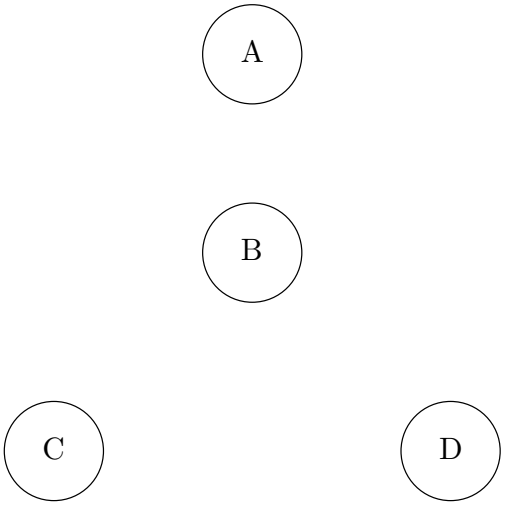

Figure 3.3: Intersection of the original networks

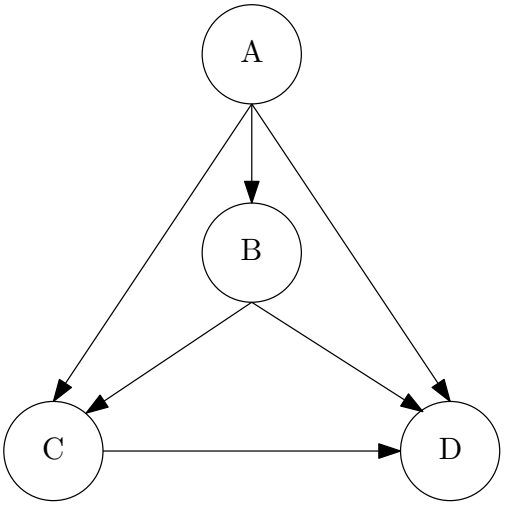

Figure 3.4: Union of the original networks

In the second article [15], it is stated that this is not sufficient for merging networks and therefore introduces another method, the output of which is the structure of the network, which can be somewhere between intersection and union. Consider still the original networks in 3.1 and 3.2. Then the result of the merged network can be, for example, Figure 3.5.

It also proposes a method for calculating the CPT of a node at different parents, e.g. node D in Figure 3.4 has different parents than both input networks (Figure 3.1 and 3.2).

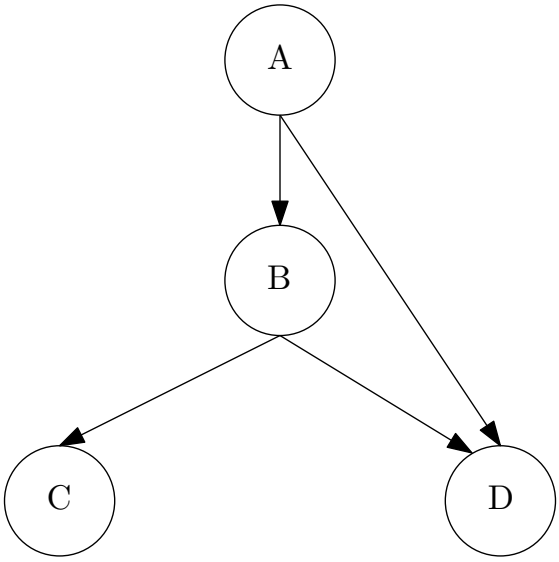

Figure 3.5: A simple Bayesian network.

Another article [16] dealing with merging is in the field of medicine. Initially, three networks were created (each from a different medical field) and the goal was to connect them. The authors mentioned the above-mentioned requirements, but then they did not connect the networks, but rather searched for the ideal structure of the Bayesian network (see section 2.5).

3.1 Differences between current articles and this thesis

In this sub-chapter, we would focus on the basic differences of previous research in the field of merging algorithms compared to the merging algorithm, which is the subject of this dissertation thesis.

The first basic difference is that the above articles do not work with data when entering the algorithm. In the beginning, they take as inputs only two (or more) Bayesian networks, exactly by definition (two points in section 2.1) and then combine them only on the basis of their structure or, where appropriate, the CPT. In contrast, our algorithm uses a score calculation (section 2.6) for each node on the input, and this score is based on (input) data.

The articles also work with different sets of nodes in the original networks. They can be the same, but also different. For our algorithm, the set of nodes for both input networks must be the same.

Another difference is that we do not use ancestral ordering to select a node to the merged network, but we use the score calculated to evaluate the nodes in order to select the nodes in the merging network.

The above-mentioned differences are based on the fact that the articles mainly try

to comply with the above-mentioned requirements. However, the aim of this work is another optimization of Bayesian networks. We do not want to preserve conditional independence from the input networks, but to find out whether the combination of these conditional independencies leads to results that are more in line with the data.

Chapter 4

Merging algorithm

We now have everything at our disposal to explain the main subject of this work, i.e. the **merging algorithm**. The algorithm is described in two ways: in general (this chapter) and mathematically (the next chapter 5). The general method is fully sufficient for its understanding, but it does not contain a deeper mathematical notation.

While reading so far, we have encountered the merging algorithm several times in the text. We said that its main benefit is **the creation of a merged network, which should provide better results for data analysis**. What exactly the word better means? We will deal with it in the further work, especially in the validation of the Bayesian network (chapter 5.7).

The algorithm is described by points that follow each other unless otherwise stated. It starts at point 1, and as soon as it returns to this point, we say that one of its **iterations** has taken place.

Now to the very description of the algorithm. Let us start with a description of its inputs.

4.1 Inputs

The Merging Algorithm needs the following inputs to work properly. The inputs consist of:

- Two input Bayesian networks according to the definition (see section 2.1). This means that each network must contain:
  - a structure,
  - conditional probability tables (also called local probability functions) assigned to each node in the structure.
- Score values for all nodes in the input networks obtained by a chosen score methods (section 2.6).
- Training data set corresponding to the nodes in Bayesian network structure.

As mentioned earlier, the algorithm requires that the sets of nodes be identical in both networks.

4.2 Remarks

Before we begin the description of the algorithm, we will first mention a few remarks that are related to the whole example:

- The issue of CPTs is not covered in order not to complicate this example. Therefore CPTs are overall omitted. We will return to it in a separate section 5.6.
- We will also not need the training data set this time. This is because it is only necessary for node score, CPT counting and also for evaluating Bayesian networks.
- The algorithm does not depend on the chosen node score method. For the score nodes only have to hold that **the lower the value of the node score, the better the data set fits on that node** (this applies to both methods mentioned in section 2.6).
- During the description of the algorithm, we will call the input networks **first** and **second** and the network that will be the result after the end of the algorithm will be called **final** in the process.
- Nodes will be marked in capital letters.
- Final Network consists of a set of nodes and a set of edges. In the beginning, these sets are empty. While the Algorithm runs, nodes and edges are subsequently added.
- **The example for each step does not have to be connected with another one.** If this happens, it is highlighted at the point. (The example in the 2nd step is based on another network structure than in the 8th step.)
- Each step of the algorithm is divided into two points
  - a description of the step,
  - an example of the step.

4.3 Description

We will start with an example, which we will use in the first few steps of the algorithm:

- The structures of the first and second Bayesian networks are shown in Figures 4.1 and 4.2.
- The score values are listed next to the respective node and also in table 4.1, where the difference in score between nodes is also calculated. We are not interested in their origin, they are here only for the purposes of this example.

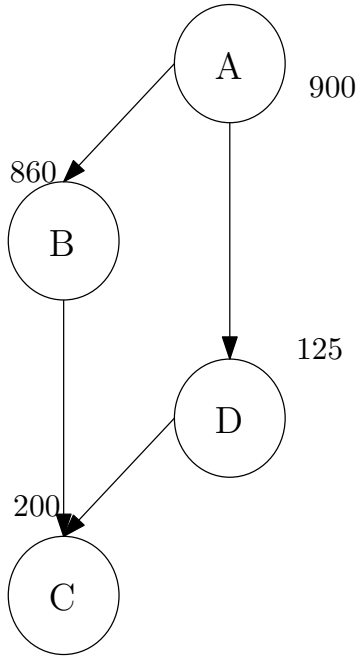

Figure 4.1:
First Bayesian Network

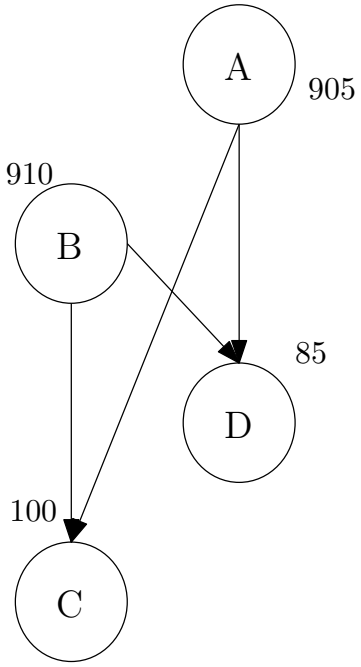

Figure 4.2:
Second Bayesian Network

Table 4.1: Score difference for each node of input Bayesian Networks

Node	Score value first network	Score value second network	Score difference
A	900	905	5
B	860	910	50
C	200	100	100
D	125	85	40

1st step

- Description

  A node is selected based on the score value differences.

- Example:

  This example is based on the score differences (Table 4.1) and Bayesian networks depicted in Figures 4.1 and 4.2). We need to find the highest difference in Table 4.1. It is the score difference 100 by the node C. Therefore, the selected node will be in this case node C.

2nd step

- Description:

  A Bayesian network is selected based on the score value of the selected node from 1st step.

- Example:

  We continue with the selected node C from the previous point. We are looking for a smaller value between the score values in Table 4.1 of node C. The score value is smaller in the second Bayesian Network (value 100). Therefore, the second Bayesian network is selected.

3rd step

- Description:

  The selected node from the 1st step is added into the Final Bayesian network if possible.

- Example:

  The selected node C will be added in the final node set.

  Note: There is also a possibility that node C is a part of the final node set. In that case, nothing happens, the final node set will not change in this step.

4th step

- Description:

  The input edges of the selected node in the selected network are going to be found.

- Example 1:

  If we continue in our previous example, then the selected node is node C and we are looking for the edges leading to the this node. This situation is depicted in Figure 4.3. There are two edges leading to node C and are highlighted in red. Hence, the result will be edges (A, C) and (B, C).

- Example 2:

  Assume the same structure and the selected node A in this network (Figure 4.3). In this case, node A has no parents. Thus, the result of this step would be an empty set of edges.

  Note: It is important to not forget the selected network (in our case second network). In mathematical notation, this is solved by the upper indexes of the nodes and/or edges.

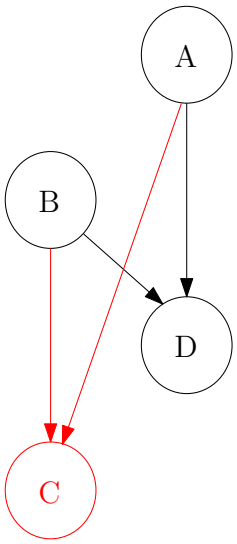

Figure 4.3: Second network with node C and edges leading to it highlighted

5th step

- Description:

  The parents of the selected node from the 1st step are going to be found.

- Example 1:

  Assume the same network structure and the selected node C. Then, the output from this step will the node set including nodes A and B.

- Example 2:

  Assume the selected node A and the Bayesian network structure depicted in Figure 4.3). Node A does not have any parents. The result will be an empty node set.

6th step

- Description:

  This step does not contain any specific action, but divides the algorithm into two branches. Which branch the algorithm goes depends on the result of 3rd step. There are two options:

  – The selected node was successfully added into the final node set in the 3rd point. → The algorithm continues with the 10th step.

  – The selected node could not be added to the set because it is already in it. → The algorithm continues with the 7th step.

- Example:

In the case of the selected node C from the 1st step, the node was successfully added in the 3rd point. Hence, the algorithm continues with the 10th step. If it was not, the algorithm would continue with the 7th step.

7th step

- Description:

  In this step we use the output from 5th step (parents of the selected node) and find out if they are present in the final node set. If they are not present, we continue with the 10th step, else with the 8th step.

- Example (not associated with the previous ones):

  To show this step on an example, we need to make assumptions:

  - the selected node is in the final node set. In the case of this example, we consider the selected node S,
  - the node S has parents (nodes) T, U, and V in the selected Bayesian network (the situation is depicted in Figure 4.4).
  - the state of the final network is depicted in Figure 4.5

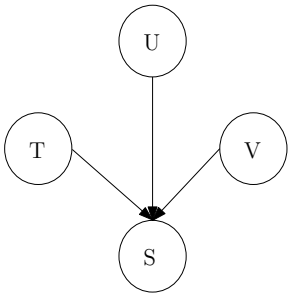

Figure 4.4: Node S in the selected network

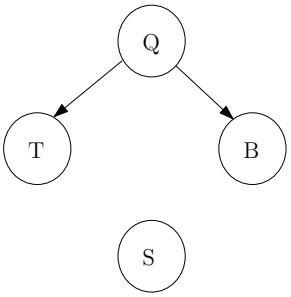

Figure 4.5: Node S in the final network

And now we compare the parent set of the selected node with the set of nodes that are contained in the final network

- Parent node set: {T, U, V}
- Final node set {Q, T, B, S}

We see that there is one node that is contained in both sets. Therefore, the algorithm would proceed to the 8th step in this state.

If the intersection of the sets were empty, the algorithm would proceed to 10th point.

8th step

- Description:

  At this point, we have reached a situation where we must verify that simply adding the selected node and its edges with parents in the final node set will not cause a cycle in the final network. We will use the set of edges from the final network and the set of edges obtained in the 4th point.

  If we find out that we would create a cycle in our final network, our algorithm will continue with 9th point. Otherwise we will continue with 10th point[1].

- Example (not associated with the previous ones):

  In this example, we want to demonstrate the origin of such a cycle. Again, we introduce some assumptions:

  – the selected Bayesian network includes edges (A, B), (A, C), (B, D), the selected network is depicted in Figure 4.6,

  – the final edges set include edges (B, C), (D, C), (C, A), the current state of the final network is depicted in Figure 4.7,

  – the selected node is node B.

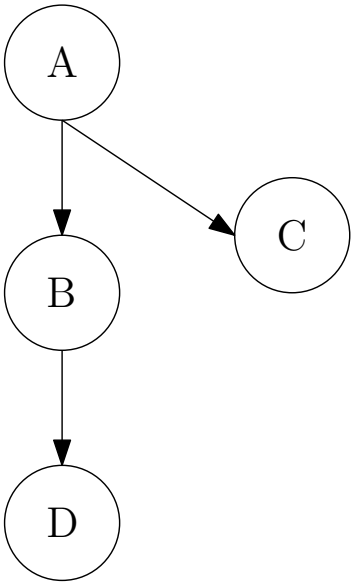

Figure 4.6: The example of the selected network

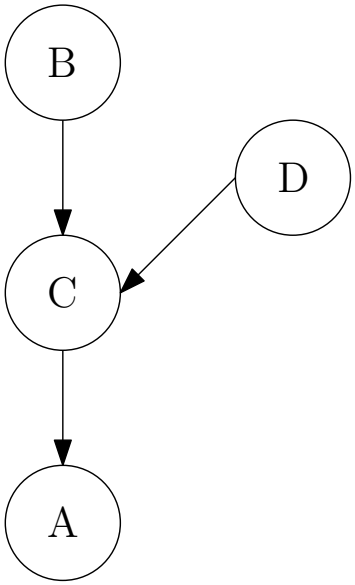

Figure 4.7: The example of the current state of the final network

[1]At this time, we will not describe an algorithm for determining cycles in a oriented graph. We recommend literature [18] for those who are interested.

We want to check the Final network on cycles. In the other words, we need to put the edges leading to node B into the Final network and check if there is any cycle present. The result for this example is depicted in Figure 4.8.

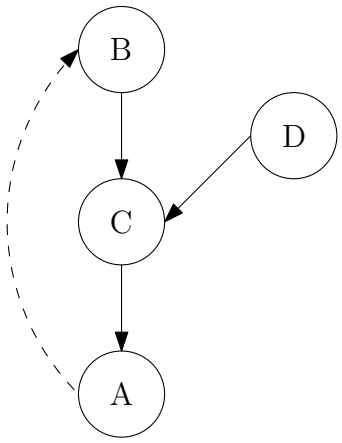

Figure 4.8: Possible feedback (B-D-C-B) found adding the edge from the selected network

We can see that adding the edge from the selected network into the Final network leads to the cycle (B-C-A-B). In this case, the algorithm will continue with the 9th step.

If there were no cycles found, Algorithm continues with the 10th step.

Subchapter 5.5 deals with the issue of cycles in the Bayesian network in more detail.

9th step

- Description:

  We have reached a situation where it is not possible to add a node and edges with its parents to the final network because a cycle would be created in the final network. The algorithm now proceeds by trying to add a node and parent edges from the other possible network (as far as this description is concerned, it means returning to 3rd point, with the other being considered the selected Bayesian network).

  If the algorithm reaches this step again, it only adds the selected node to the network (if it is not already in it) and does not add the edges with the parents to the final network. A separate section 5.5 is devoted to the issue of cycles.

- Example:

  In the case of the example from the previous point (Figure 4.8), the edge is not added into the final network, the current state of the final network remains the same (depicted in Figure 4.7) and the algorithm continues with the 1st point.

10th step

- Description:

  In this step, we will add the parents of the selected node and the edges leading to it to the final network.

- Example (not associated with the previous ones):

  The example of this step is based on the following networks:

  – The selected Bayesian network is depicted in Figure 4.9.
  – The actual state of the Final network is depicted in Figure 4.10.
  – The selected node is node C.

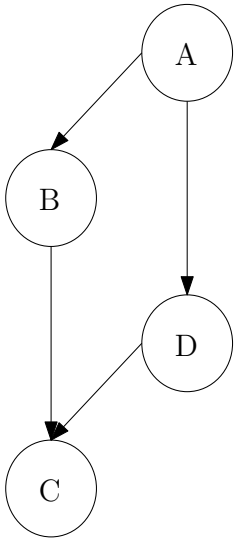

Figure 4.9: The selected Bayesian network

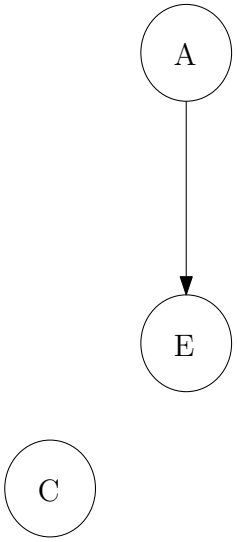

Figure 4.10: The example of the current state of the final network

The algorithm adds all the parents and relevant edges leading to the selected node from the selected network into the Final network. The result is depicted in Figure 4.11.

11th step

This step involves assigning the CPT of the selected node to the final network. There are basically two options based on whether a CPT from the selected network can be used. If possible, we assign a CPT to the node and continue to the last 12th point of the algorithm. If this is not possible (due to the creation of a cycle in the algorithm), then the CPT must be calculated. We talk more about this issue in the section 5.6.

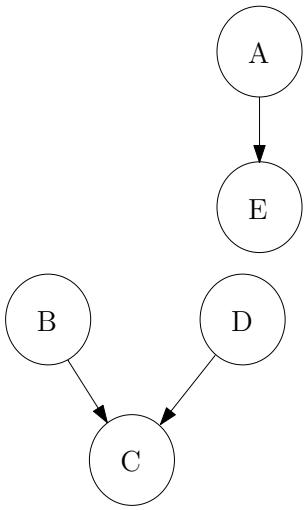

Figure 4.11: The state of the final network after adding the parents and edges from the selected network

12th step

We have now performed all possible operations with the selected node and therefore the algorithm continues to step 1. If all nodes have already been selected, the algorithm ends.

Chapter 5

Mathematical description of Merging Algorithm

This chapter focuses on the description of the algorithm more in the mathematical level. To understand its operation, we recommend knowing the basics of graph and set theory.

General information on the algorithm can be found at the beginning of the previous chapter 4.

Let us start with the mathematical notation of inputs.

5.1 Inputs into the Algorithm

The algorithm must have the following inputs:

- two input Bayesian networks (we called them first and second again) according to the definition,
- node scores assigned to each node,
- a training data set.

As already mentioned, the Bayesian network consists of a structure (nodes and edges) and a CPT assigned to each node. The data set must correspond to the input Bayesian networks, i.e. the variables in the data set must correspond exactly to the nodes in the input Bayesian networks.

The data set generally contains n variables

$$\mathbf{X} = \{x_1, x_2, \ldots, x_n\}. \tag{5.1}$$

During the description of the algorithm we can use the word node or variable, in this context they have the same meaning.

The input Bayesian networks have the same set of nodes $\mathbf{X}$ against a set of edges that differs.

The first Bayesian Network is defined by a set of nodes $\mathbf{X}$ and a set of edges

$$\mathbf{E}^{(1)} = \{E^{(1)}_{1,s,t}, E^{(1)}_{2,s,t}, \ldots, E^{(1)}_{a,s,t}\}, \tag{5.2}$$

where a means the number of all edges (relations) in the first Bayesian Network, s denotes an index of a source node and t index of a target node on a particular edge.

We have also introduced a superscript (1) here, which will always indicate that a set or a vector belongs to the first Bayesian network. For the second Bayesian network, the superscript will be (2).

This second network is then defined by the same set of nodes $\mathbf{X}$ and a set of edges

$$\mathbf{E}^{(2)} = \{E^{(2)}_{1,s,t}, E^{(2)}_{2,s,t}, \ldots, E^{(2)}_{b,s,t}\}, \tag{5.3}$$

where b means the number of all edges (relations) in the second Bayesian Network.

At this moment, Bayesian network structures for both input networks are successfully defined.

5.1.1 Sets and vectors assigned to nodes

Each node in both input networks has its conditional probability mass function assigned. For Algorithm purposes, sets of these conditional probability functions are defined. The set of probability functions $\mathbf{F}^{(1)}$ is defined

$$\mathbf{F}^{(1)} = \{f^{(1)}_{x_1}, f^{(1)}_{x_2}, \ldots, f^{(1)}_{x_n}\}, \tag{5.4}$$

for the first Bayesian network and the set $\mathbf{F}^{(2)}$

$$\mathbf{F}^{(2)} = \{f^{(2)}_{x_1}, f^{(2)}_{x_2}, \ldots, f^{(2)}_{x_n}\}, \tag{5.5}$$

for the second Bayesian network.

Beyond the definition of the Bayesian network, we assign a score to each node in both networks. We have already talked about it in the section 2.6 and now we will use it. The vector of scores $\mathbf{S}^{(1)}$ is denoted

$$\mathbf{S}^{(1)} = [s^{(1)}_{x_1}, s^{(1)}_{x_2}, \ldots, s^{(1)}_{x_n}] \tag{5.6}$$

for the first Bayesian network and the vector of scores $\mathbf{S}^{(2)}$ is accordingly denoted

$$\mathbf{S}^{(2)} = [s^{(2)}_{x_1}, s^{(2)}_{x_2}, \ldots, s^{(2)}_{x_n}]. \tag{5.7}$$

for the second Bayesian network.

The last vector defined in this section includes absolute values of score differences computed from vectors $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$ for each node in $\mathbf{X}$,

$$\mathbf{S} = [s_1, s_2, \ldots, s_n] = [|s^{(1)}_{x_1} - s^{(2)}_{x_1}|, |s^{(2)}_{x_2} - s^{(2)}_{x_2}|, \ldots, |s^{(1)}_{x_n} - s^{(2)}_{x_n}|]. \tag{5.8}$$

5.2 Final Network

The main output of the Merging Algorithm is the Final Bayesian Network with all its components (structure and CPTs). Before the Algorithm starts, the Final Bayesian Networks is defined as an empty set of nodes

$$\mathbf{X}^{(m)} = \{\}, \tag{5.9}$$

an empty set of edges

$$\mathbf{E}^{(m)} = \{\}, \tag{5.10}$$

and an empty set of conditional probability functions (CPTs)

$$\mathbf{F}^{(m)} = \{\}. \tag{5.11}$$

While the Algorithm runs, the sets in Final Bayesian Network will be subsequently filled by nodes, edges, and conditional probability functions.

5.3 What has been defined until now?

In the previous sections, we introduced many different sets and vectors. Let us repeat them all for clarity.

Bayesian network 1

A set of nodes: $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ (same for Bayesian network 2).
A set of edges: $\mathbf{E}^{(1)} = \{E^{(1)}_{1,s,t}, E^{(1)}_{2,s,t}, \ldots, E^{(1)}_{a,s,t}\}$.
A set of conditional probability functions assigned to each edge: $\mathbf{F}^{(1)} = \{f^{(1)}_{x_1}, f^{(1)}_{x_2}, , \ldots, f^{(1)}_{x_n}\}$.
A vector of scores: $\mathbf{S}^{(1)} = [s^{(1)}_{x_1}, s^{(1)}_{x_2}, \ldots, s^{(1)}_{x_n}]$.

Bayesian network 2

A set of nodes: $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ (same for Bayesian network 1).
A set of edges: $\mathbf{E}^{(2)} = \{E^{(2)}_{1,s,t}, E^{(2)}_{2,s,t}, \ldots, E^{(2)}_{b,s,t}\}$.
A set of conditional probability functions assigned to each edge: $\mathbf{F}^{(2)} = \{f^{(2)}_{x_1}, f^{(2)}_{x_2}, , \ldots, f^{(2)}_{x_n}\}$.
A vector of scores: $\mathbf{S}^{(2)} = [s^{(2)}_{x_1}, s^{(2)}_{x_2}, \ldots, s^{(2)}_{x_n}]$.

Final network

A blank set of nodes: $\mathbf{X}^{(m)} = \{\}$.
A blank set of edges: $\mathbf{E}^{(m)} = \{\}$.

A blank set of conditional probability functions: $\mathbf{F}^{(m)} = \{\}$.

There is also score vector $\mathbf{S}$ that does not belong to any of the networks: $\mathbf{S} = [s_1, s_2, \ldots, s_n]$.

5.4 Algorithm Description

The Algorithm will be described by an order of points. Unless otherwise stated, at each point the algorithm proceeds to the next point.

1. Choose a node that was not already chosen from $\mathbf{X}$.

   We select a node based on the difference between node scores. The highest difference is always selected. Mathematically, we look for the index with the highest value from the vector $\mathbf{S}$ and then store it in the variable i

   $$i = \text{argmax}(\mathbf{S}). \tag{5.12}$$

   This node will not be selected again, so we will set the selected value to zero,

   $$s_i = 0. \tag{5.13}$$

   The selected node is denoted x_i.

2. Choose input Bayesian Network

   We also need to choose an input Bayesian network. This is done by comparing the respective score values in the vectors $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$. If this equation applies

   $$s_{x_i}^{(1)} > s_{x_i}^{(2)} \tag{5.14}$$

   then the second Bayesian network is chosen (the upper index (2)). Otherwise, the first Bayesian network is selected (the upper index (1)). The index of the selected Bayesian network is then stored in the variable j:

   $$j = \begin{cases} 1 & s_{x_i}^{(1)} < s_{x_i}^{(2)}, \\ 2 & s_{x_i}^{(1)} > s_{x_i}^{(2)}. \end{cases} \tag{5.15}$$

3. Add the selected node x_i into Final Network

   First, we need to find out if the node x_i is present in the final node set $\mathbf{X}^{(m)}$. Next, we introduce the variable *add*

   $$add = \begin{cases} 0 & x_i \in \mathbf{X}^{(m)}, \\ 1 & x_i \notin \mathbf{X}^{(m)}. \end{cases} \tag{5.16}$$

Its value will be equal to false (0) if the selected node x_i is already present in the final node set $\mathbf{X}^{(m)}$. Otherwise, the variable *add* is set to true (1). The *add* variable will be useful in later steps of the algorithm.

Then the selected node x_i is either added into the set $\mathbf{X}^{(m)}$ or nothing happens.

$$\mathbf{X}^{(m)} = \begin{cases} \mathbf{X}^{(m)} \cup \{x_i\} & x_i \notin \mathbf{X}^{(f)}, \\ \mathbf{X}^{(m)} & x_i \in \mathbf{X}^{(f)}. \end{cases} \tag{5.17}$$

4. Search the input edges of the selected node x_i

   In this step of the algorithm, we will be interested in the edges that lead to the node x_i. Before we begin to describe this whole operation mathematically, let us remember the marking of edges. Generally the edge

   $$E^{(j)}_{k,s,t} = (x^{(j)}_{s,k}, x^{(j)}_{t,k}) \tag{5.18}$$

   is in the j-th Bayesian network and can be rewritten as an ordered pair from a source node $x^{(j)}_{s,k}$ to a target node $x^{(j)}_{t,k}$ where s denotes an index of a source node and t an index of a target node. The variable k denotes the order of the edge in the set.

   So we are looking for edges from the edge set $\mathbf{E}^{(j)}$ in j-th Bayesian network that satisfy the condition that the target node is the selected node x_i and we then store them in the set $\mathbf{H}$.

   Mathematically,

   $$\mathbf{H} = \{E_{k,s,t} | E_{k,s,t} \in \mathbf{E}^{(j)} \wedge k = 1, .., \mathrm{n}\left(\mathbf{E}^{(j)}\right) \wedge\ t = i\ \wedge\ s = 1, .., n\,\}. \tag{5.19}$$

   Let us analyze the above complex notation. Generally, it consists of five parts:

   - $E_{k,s,t}$ – means that we are looking for the edges
   - $|$ – the vertical line separates the conditions that must be met
   - $E_{k,s,t} \in \mathbf{E}^{(j)}$ – resulting edges must be from the selected Bayesian network (the variable j)
   - $k = 1, .., \mathrm{n}\left(\mathbf{E}^{(j)}\right)$ – k is going from 1 to the set cardinality
   - $t = i$ – the key condition, the index t of a target node in $E_{k,s,t}$ must be equal to the index i of the selected node x_i
   - $s = 1, .., n$ – the index of the source node must be between 1 and the number of all edges in the set $\mathbf{X}$

5. Find parents of the selected node x_i

We already have the edges leading to the selected node in the set **H**. The next operation is to find out the parents of the selected node. In other words, we need to select the source nodes from the set **H**.

$$\mathbf{P} = \{x_p | x_p \in \mathbf{X} \wedge (E_{k,s,t} \in \mathbf{H} : p = s)\}. \tag{5.20}$$

This notation goes through all nodes in the set **X** and selects those that satisfy the condition that the index of the source node s is equal to the index p of the node in the set **X**. The relevant nodes are then saved to the set **P**.

6. Algorithm branching based on the value of the *add* variable

   We introduced the variable *add* in point 3 in order to find out whether the selected node x_i is present in the final node set $\mathbf{X}^{(m)}$.

   Based on its value, the algorithm then branches

$$add = \begin{cases} 0 & \text{go to the point 7,} \\ 1 & \text{go to the point 10.} \end{cases} \tag{5.21}$$

   The principle of this branching is simply that if the selected node x_i is not yet in the final node set $\mathbf{X}^{(m)}$, then a cycle cannot occur.

7. Find parents of the selected node x_i present in the set $\mathbf{X}^{(m)}$

   We are currently in the branch where the selected node x_i is present in the final node set $\mathbf{X}^{(m)}$. Now we need to find out the presence of the parents of the selected node x_i in the set $\mathbf{X}^{(m)}$. We do this by intersecting sets $\mathbf{X}^{(m)}$ and **P**. The result will be stored in the set **T**.

$$\mathbf{T} = \mathbf{X}^{(m)} \cap \mathbf{P} \tag{5.22}$$

   If the set T is empty, it means that no parents of the selected node x_i are in the final node set $\mathbf{X}^{(m)}$ and therefore a cycle cannot be created. Hence, the algorithm will continue with point 10. Otherwise it will continue with point 8. Mathematically,

$$\mathbf{T} = \begin{cases} \emptyset & \text{go to the point 10,} \\ \neg\emptyset & \text{go to the point 8.} \end{cases} \tag{5.23}$$

8. Find cycle

   Unfortunately, we now risk that by adding the selected node with its parent links, we will cause a cycle in the final network. We will therefore prepare a set **U** of all relevant edges in which we will search for the cycle. It will consist of the set of edges of the final network $\mathbf{E}^{(m)}$ and the set of edges between the selected node and its parents **H**. Mathematically written

$$\mathbf{U} = \mathbf{E}^{(m)} \cup \mathbf{H}. \tag{5.24}$$

We then apply a cycle search algorithm to this set.

Searching for cycles in oriented graphs is a separate chapter of graph theory and we will not deal with it here. For completeness, we present resources [18] that deal with this issue.

All we need here is information on whether the set $\mathbf{U}$ contains a cycle. We introduce the auxiliary variable *cycle*. Its value will depend on this information as follows

$$cycle = \begin{cases} 0 & \mathbf{U} \text{ does not include cycle, go to 10,} \\ 1 & \mathbf{U} \text{ includes cycle, go to point 9.} \end{cases} \tag{5.25}$$

9. New setup while the same selected node x_i

   We reached the point where we cannot add the edges between the selected node x_i and its parents due to cycle.

   It is important how this point is reached. If it is the first time for the selected node x_i, there is still one more option here: to select the other input Bayesian network although the selected node x_i has the worse score in it. In the other words the variable j changed:

   $$j = \begin{cases} 1 & j = 2, \\ 2 & j = 1. \end{cases} \tag{5.26}$$

   The temporary sets also need to be cleared, the variable *add* is set to false (the selected node x_i is already present in the set $\mathbf{X}^{(m)}$):

   $$\begin{aligned} &\mathbf{P} = \emptyset \\ &\mathbf{H} = \emptyset \\ &\mathbf{U} = \emptyset \\ &\mathbf{T} = \emptyset \\ &add = 0(\text{false}) \end{aligned} \tag{5.27}$$

   and the algorithm continues with point 4.

   Unfortunately, if this is the second time (the variable j was set to both values 1 and 2 while the selected node x_i does not change) and there is still a cycle in the final network, the algorithm will leave the node as it is. This means that it remains in the final set of nodes $\mathbf{X}^{(m)}$ and the algorithm continues at 12.

   Unfortunately, this solution may cause the final network to be no better than the input networks. More information about this issue can be found in a separate section 5.5.

10. Add missing parents into the final node set $\mathbf{X}^{(m)}$

    At this point, we will add the missing parents to the final node set $\mathbf{X}^{(m)}$. We have the parents of the selected node x_i available in the set P and we just need to unite with the final node set $\mathbf{X}^{(m)}$.

    $$\mathbf{X}^{(m)} = \mathbf{X}^{(m)} \cup \mathbf{P}. \tag{5.28}$$

This approach also avoids problems with the presence of a parent in both sets $\mathbf{P}$ and $\mathbf{X}^{(m)}$.

11. Add edges into the final edge set $\mathbf{E}^{(m)}$

    We will now do the same with the relevant edges (i.e. edges leading to the selected node x_i) of the selected node x_i stored in the set $\mathbf{H}$. Again, we will use the union of this set and the final edge set $\mathbf{E}^{(m)}$.

    $$\mathbf{E}^{(m)} = \mathbf{E}^{(m)} \cup \mathbf{H}. \tag{5.29}$$

12. Assignment of the conditional probability function

    So far, all points of the algorithm have solved the structure of the network. Now it is time to assign a conditional probability function to the selected node x_i.

    This is based on the previous point of the algorithm:

    - The previous point is point 11

      In this case, the selected node and its input edges correspond to the node and the input edges from the selected Bayesian network. Thus, the conditional probability function from the selected Bayesian network $f_{x_i}^{(j)}$ is simply added into the final set of conditional probability functions $\mathbf{F}^{(m)}$:

      $$\mathbf{F}^{(m)} = \mathbf{F}^{(m)} \cup \{f_{x_i}^{(j)}\} \tag{5.30}$$

    - The previous point is point 9

      In this situation, we do not have a chance to assign a conditional probability function directly from the input Bayesian networks. Section 5.6 deals with the issue of calculating conditional probability functions of a selected node when it is not possible to take any of the input networks.

      For this description, it is enough to somehow obtain the conditional probability function f_{x_i} and then assign it to the selected node x_i. This function is then added into the final set of conditional probability functions $\mathbf{F}^{(m)}$:

      $$\mathbf{F}^{(m)} = \mathbf{F}^{(m)} \cup \{f_{x_i}\} \tag{5.31}$$

13. Setup for next iteration

    Now, we have done everything that was needed with the selected node and will perform the setup. This means that we set all variables and auxiliary sets to

default values

$$\begin{aligned}
&\mathbf{P} = \emptyset,\\
&\mathbf{H} = \emptyset,\\
&\mathbf{U} = \emptyset,\\
&\mathbf{T} = \emptyset,\\
&add = 1,\\
&cycle = 0,\\
&i = 0,\\
&j = 0.
\end{aligned} \tag{5.32}$$

If the vector of score value differences **S** includes only the zero values

$$s_i = 0, \quad i = 1, \ldots, n \tag{5.33}$$

then the algorithm ends. Otherwise, the algorithm proceeds to point 1.

5.5 Cycles

When the algorithm is running, there may be situations where it is not possible to add the edge between the parents of the selected node and the selected node (point 9). This would cause a cycle in the final network. This section discusses some suggestions on how to deal with this problem in the future.

Now the implementation of the algorithm is performed as follows: if a cycle should occur, no edges between the parents and the selected node will be added to the final network and the algorithm continues.

Let us now show you a few ways to deal with this situation. We consider the following situation:

- the selected network is depicted in Figure 5.1,
- the actual state of the Final network is illustrated in Figure 5.2,
- the selected node is node B.

Ideally, the resulting network would look as shown in Figure 5.3 (Added edges are shown by a dashed line).

Unfortunately, it can be seen that adding these edges will cause a B-C-A-B cycle.

There are basically two ways to deal with this situation:

- Leave the node in the final network without adding edges (procedure now implemented in the algorithm).
- Connect as many parents to the selected node as possible.

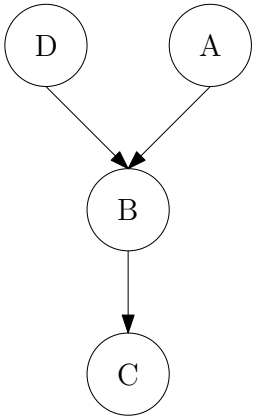

Figure 5.1: The example of the selected network

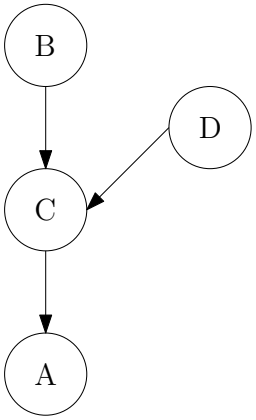

Figure 5.2: The example of the current state of the final network

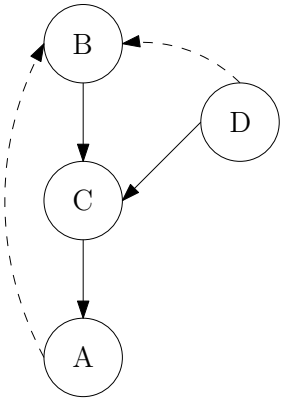

Figure 5.3: The resulting network including both edges from the selected network.

- Connect the most suitable combination of different parents of the selected node.

In general, if we get into this situation, none of these options will guarantee us that the final network will be better than the input networks. We can try to prevent this as much as possible.

However, all of the above options require counting the node score while the algorithm is running. The principle is that we would calculate the node score for a situation where we do not add any edge. Next, we would calculate the scores for all combinations of acceptable edges. We would then compare these values and then select the one from the best evaluation in the final network.

This is one of the issues that remains as an opportunity to improve the algorithm in the future.

5.6 Conditional Probability Tables

Point 12 of the algorithm describes the process of assigning a conditional probability function to a selected node. If it is possible to take a conditional probability function from a conditional probability function set in the selected network, the algorithm assigns this function and continues.

If the previous process cannot be done, we must choose another approach. This situation occurs when the node cannot be placed in the final network with edges to its parents. In section 5.5 in Figure 5.2 we see that the final network contains node B without parents, and in Figure 5.3, that parents from the selected network cannot be added because a cycle would be created. In this section, we will focus on this case.

In this situation, we will use the fact that we have the training data set and perform the calculation of conditional probability functions from it.

We could easily use the EM algorithm for computing CPT [19] that is implemented in GeNie. However, we have the merging algorithm implemented outside GeNie and therefore we must use a different approach. We will use a calculation based on the relative frequencies of the values at this time.

5.6.1 Calculation based on relative frequencies

We will show this calculation on a simple example. Suppose the structure depicted in figure 5.4 and these assumptions:

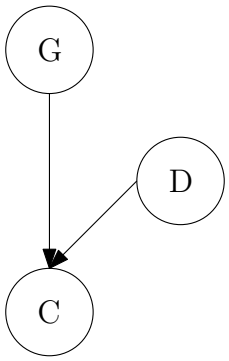

Figure 5.4: The node C with its parents

- We will compute the conditional probability function for node C in the Final Network because it cannot be obtained from the input network.
- Node C and its parents have only two values
- The data records are listed in Table 5.1

Mathematically, we are looking only for the probability function $P(C|G, D)$. The computation begins with the counting of the relevant frequencies, e.g. G=0, D=0 and

Node G	Node D	Node C
1	0	1
1	1	1
0	1	0
0	1	1
1	1	0

Table 5.1: The sample training data set for computing CPT

C=0 occurs in data table 5.1 only once and therefore the value in table 5.2 in the appropriate cell is 1. All frequencies are in Table 5.2.

Node G	Value 0		Value 1	
Node D	Value 0	Value 1	Value 0	Value 1
Node C - Value 0	0	1	0	1
Node C - Value 1	0	1	1	1

Table 5.2: The frequencies based on the Table 5.1

Now we only need to normalize these frequencies in each column and we get the required conditional probability function $P(C|G, D)$. If there is a situation where the parent combination is not included in the data (e.g. the first column in Table 5.2 for G = 0, D = 0), the probabilities of the selected node are distributed uniformly.

Node G	Value 0		Value 1	
Node D	Value 0	Value 1	Value 0	Value 1
Node C - Value 0	0.5	0.5	0	0.5
Node C - Value 1	0.5	0.5	1	0.5

Table 5.3: Probability function $P(C|G, D)$, CPT of the node C after normalization process

It is clear that for a larger number of data records it would be necessary to use at least the algorithm implemented in GeNie. For the purposes of this work, counting based on relative frequencies is sufficient.

5.7 Evaluation of Merging Algorithm

We have finally reached the stage where we have two input networks and a merged network. The task of this chapter is to explain the principle of the overall evaluation of the algorithm. In other words, prove that the merged network (i.e. output of the algorithm) is better than the input networks.

In our case, one Bayesian network is better than the other when this Bayesian network fits a given data set more than the other.

Typically, the data set is divided into two parts (2/3 training data set and 1/3 validation data set). Training data set is also used as input to the merging algorithm.

The evaluation process of the merging algorithm then takes place in two stages:

- testing part using training data set,
- validation part using validation data set.

It should be the case that the merged network will give better evaluation results in both of these parts of the evaluation.

These parts are basically the same and differ only in the data set used, so the following procedure applies to both.

First we need to choose the method of calculating the score node and with its help the calculation of the overall evaluation of the network. This method can be the same or different from the one we used for the merging algorithm.

We will now describe the procedure for both methods mentioned in the section 2.6.

5.7.1 Likelihood-based method

As a reminder, the node score using the likelihood-based approach is calculated as the sum of the logarithms of the respective parameters

$$\ln\left(L_N(\hat{\Theta})\right) = \sum_{n=1}^{N} \ln\left(\hat{\Theta}_{y_n|\mathrm{Pa}(y)}\right). \tag{5.34}$$

The explanation of the variables and more detailed information about this method can be found in the section 2.6.2.

It is important for us that the equation 5.34 is used to calculate the score of all nodes in a Bayesian network. Then, we sum up these calculated scores to obtain the final network score

$$L = \sum_{y=1}^{Y} \sum_{n=1}^{N} \ln\left(\hat{\Theta}_{y_n|\mathrm{Pa}(y)}\right). \tag{5.35}$$

where Y is the number of all nodes in the Bayesian network.

This L value is calculated for all three networks (two input and merged). **It holds that the lower the log-likelihood value of the Bayesian network, the better the data fits on that Bayesian network.**

5.7.2 Prediction error

The same is true for the predictive error method. For a specific node, its value is calculated

$$e_y = \sum_{n=1}^{N} e_n = \sum_{n=1}^{N} |y_n - \hat{y}_{\mathrm{Pa}(y)}|. \tag{5.36}$$

The explanation of the variables and more detailed information about this method can be found in the section 2.6.3.

Again, we only sum up the errors of the individual nodes in a Bayesian network

$$e = \sum_{y=1}^{Y} e_y = \sum_{y=1}^{Y} \sum_{n=1}^{N} |y_n - \hat{y}_{\mathrm{Pa}(y)}| \tag{5.37}$$

where Y is the number of all nodes in the Bayesian network.

This e value is calculated for all three networks (two input and merged). **It holds that the lower the prediction error value of the Bayesian network, the better the data fits on that Bayesian network.**

We will show the evaluation process as part of an example in the next chapter.

Chapter 6

Traffic accident analysis example

As said before, Bayesian networks are modeling tools that can be used mainly for data analysis in terms of prediction, decision making, etc. We can create many different Bayesian networks related to a defined example. Generally, the goal is always to make the best possible data analysis.

Our task will be related to the issue of traffic accidents. We will describe the data available for this task and create two Bayesian networks based on them. The first will be created by an expert and the second will be generated by a network structure algorithm (An example of such an algorithm is in the section 2.5.2). Then we merge these networks using our algorithm.

But our task in this case will not be data analysis, but **to prove that a network created by the merging algorithm is better for data analysis**.

6.1 Data description

For this example, we have traffic accident data from 2012 provided by City Hall in Prague. This data contains several types of variables:

- variables related to weather at the time of an accident: State of a surface, Weather, Wind conditions, etc.,
- variables related to a place where the accident happened: Location, Road division, Specific objects and places nearby, etc.,
- variables related to the accident itself: Main causes of an accident, Number of involved vehicles, etc.,
- damage-related variables: the number of minor, major injuries, death and material damage connected with the accident.

  Based on these damage-related variables, we calculate the accident severity variable

$$AS = 433 \cdot a + 4867,7 \cdot b + 19440 \cdot c + d. \tag{6.1}$$

where a is the number of minor injuries, b is the number of serious injuries, c is the number of deaths and d is material damage connected with the accident. Based on the numerical value of accident severity variable, we divide it into three intervals representing a light accident, a medium accident, and a serious accident.

More information on the values of all variables can be found in Appendix A.

6.2 Task definiton

Our task for data analysis in this case would be to investigate the causes of serious (or moderate) traffic accidents. On this basis, it is then possible to start creating Bayesian networks that will be suitable for this analysis.

In the following text, we will distinguish between the Accident severity variable, which we will call the output variable, and other variables that will be causal.

Now we have everything ready for the creation of Bayesian networks according to the given task. Let us start with the input expert network.

6.3 Bayesian network created by an expert

To create an expert Bayesian network, we approached an expert who understands both the issue of traffic accidents and also informatics. We explained to him the basics of Bayesian networks (Chapter 2), the task definition (section 6.2), and gave him the variables and their values.

His task was to design the structure of the Bayesian network based on his knowledge. All other text therefore corresponds to the opinions of this expert.

He divided the variables into several groups:

- the main variables, that means crucial variables,
- variables related to a driver and weather (cannot be influenced by the road administrator),
- variables related to a location,
- other variables,
- output variable: Accident severity.

There are three main variables: Main causes, Type of accident, and Type of collision. Main causes impact Type of accident which then impacts Type of collision. All the main variables impact the output variable Accident severity, see Figure 6.1.

The variables related to a driver and weather include Alcohol present, State of surface, Visibility, and Wind conditions variables. They are connected with Main causes as shown in Figure 6.2.

The other branch of the network structure is related to an accident location. It begins with the Road division that impacts Location accident and also Location crossing. The

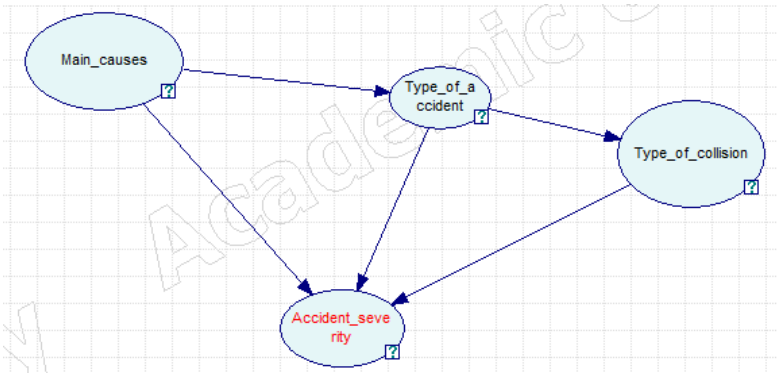

Figure 6.1: Main variables of the expert network

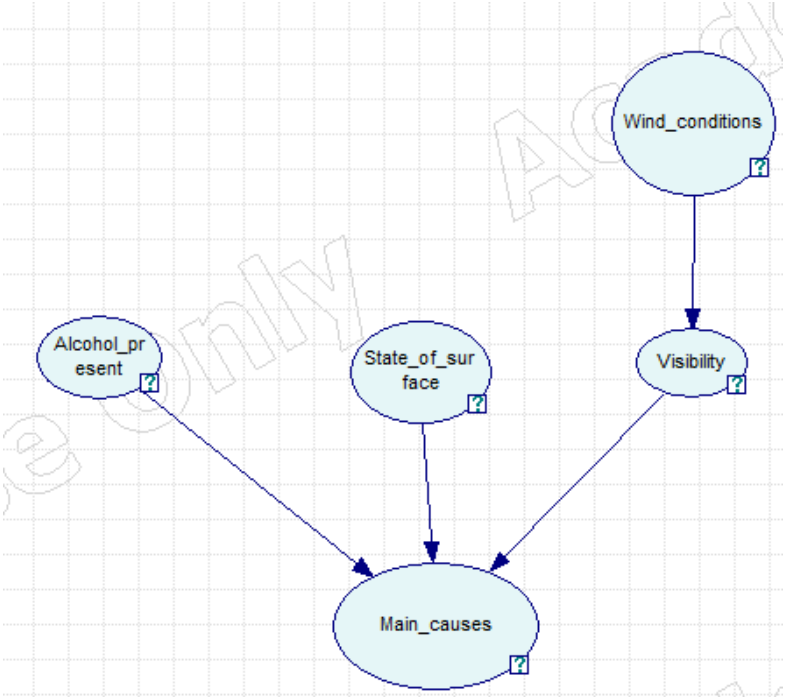

Figure 6.2: The variables related to a driver and weather and their relation to Main variables

location of an accident can be in a lane, on tracks, or elsewhere while the accident in an intersection depends on its type. For simplification, the location of the intersection can be in side streets (right preference rule) or in main streets. Road division also impacts Layout. The result of this part of the network is depicted in Figure 6.3.

Type of collision is influenced by Location of an accident and Specific places nearby if any. If an accident happens in an intersection it primarily impacts Type of collision, see Figure 6.4.

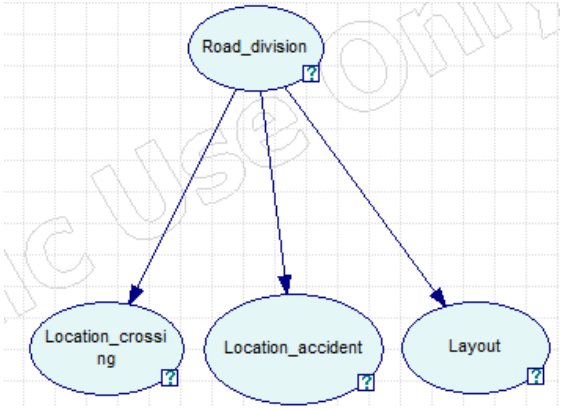

Figure 6.3: The variables related to a location of an accident, part 1 – Road division

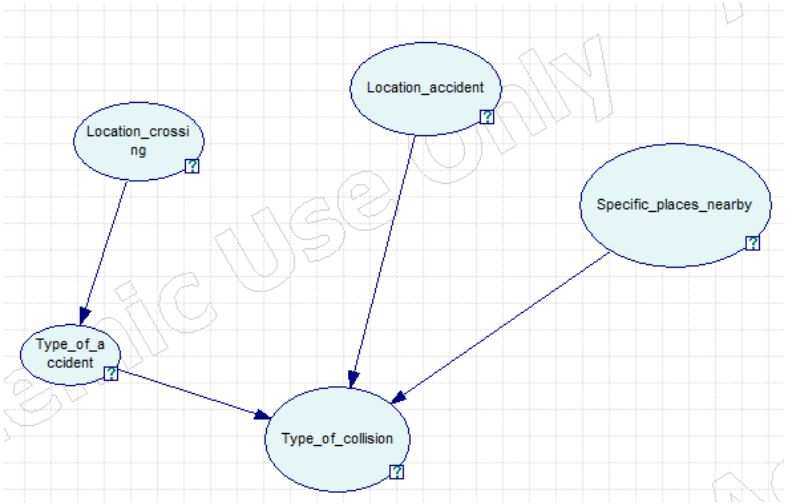

Figure 6.4: The variables related to a location of an accident, part 2 – Type of collision

The last part of the network includes the other variables: Number of participants and Type of barrier. These two variables are not related to the output variable in terms of the path between these variables and the output one. They are influenced by Type of collision, see Figure 6.5.

Having these five parts together leads to the structure of the expert network depicted in Figure 6.6.

The second part of the Bayesian network, CPTs, is computed from the data using an appropriate algorithm (in this case EM algorithm). More about this issue can be found in the section 5.6.

It is very important to realize that the whole network is designed by one expert. The other experts can have another idea of network designing and can be made in a completely different way.

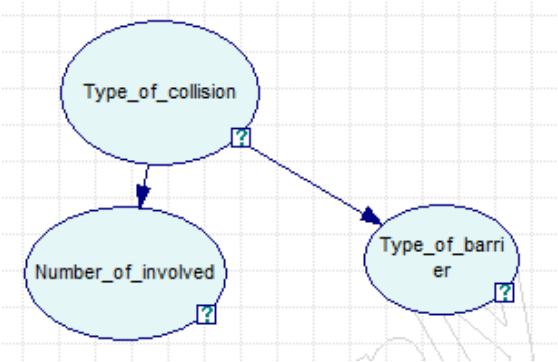

Figure 6.5: The other variables

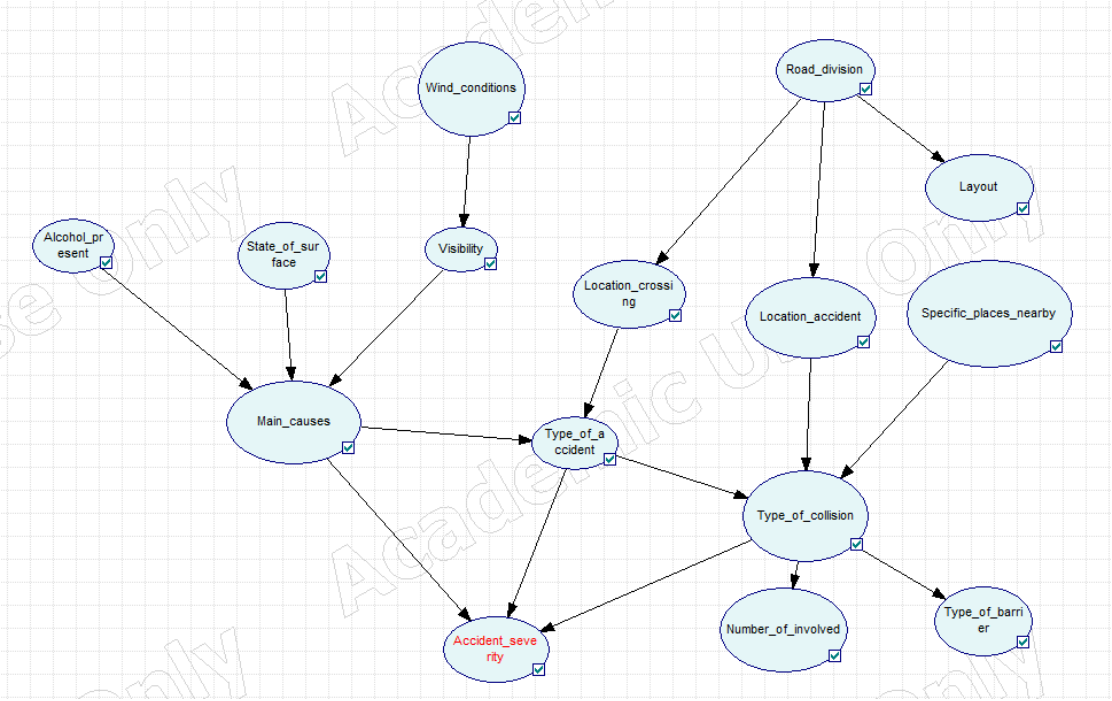

Figure 6.6: The whole expert network

6.4 Bayesian network generated by structure algorithm

The second Bayesian network is generated by a structure algorithm based on a Bayesian search principal, see section 2.5.2. The input to this algorithm is only the data from section 6.1. We set one condition for this algorithm: the accident severity node must be output, i.e. it has no children.

CPTs are calculated similarly to an expert network, by the EM algorithm.

The resulting Bayesian network structure is depicted in Figure 6.7.

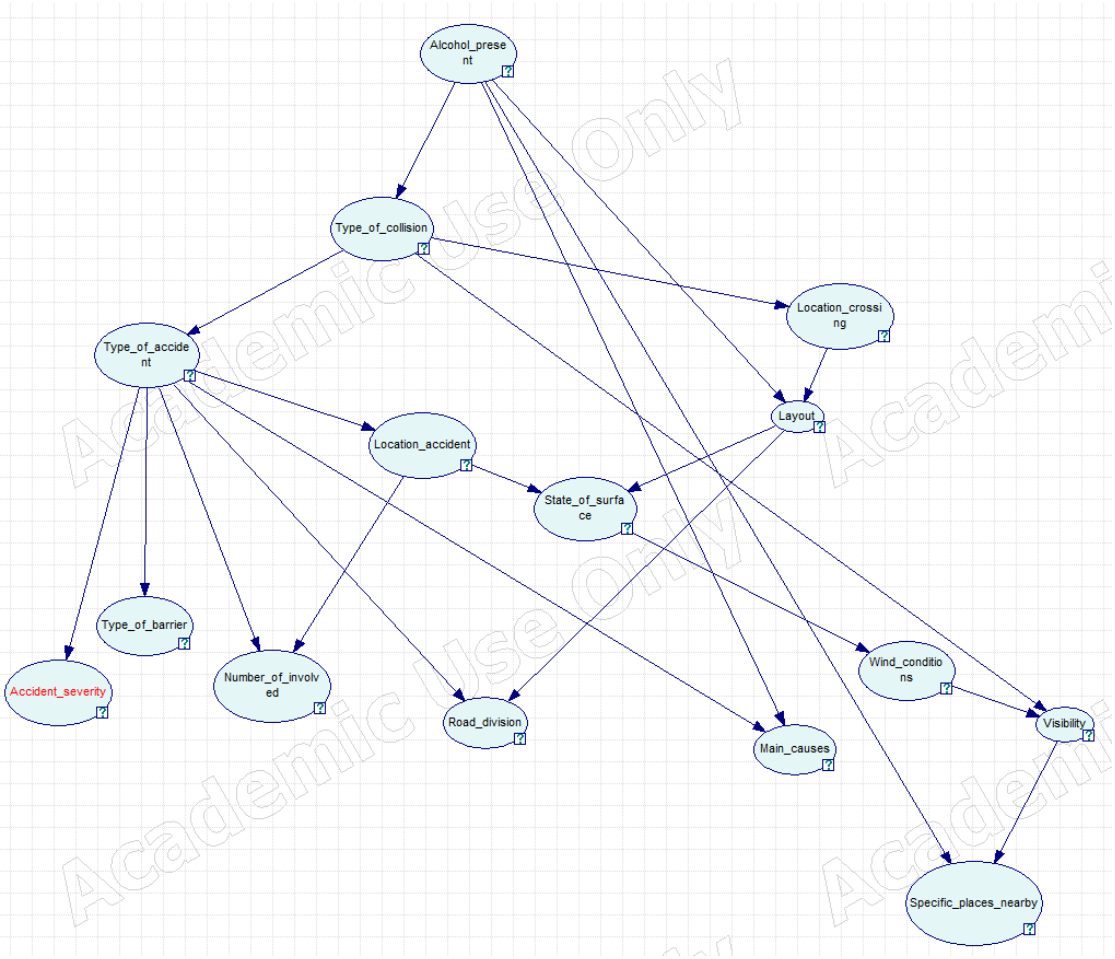

Figure 6.7: The whole algorithm network

So we now have two Bayesian networks created specifically for the task. We will now use the merge algorithm and comment on its results.

6.5 Final network

The structure of the final network is shown in figure 6.8. It differs in quite a few edges compared to input networks.

Let us now to discuss the results focusing on the main differences with the input networks.

It is not surprising that more nodes were chosen from the algorithm network. There are also several nodes selected from the expert network and one node cannot be selected from the input ones.

If we go more into detail of the Final network, we can see that the output variable

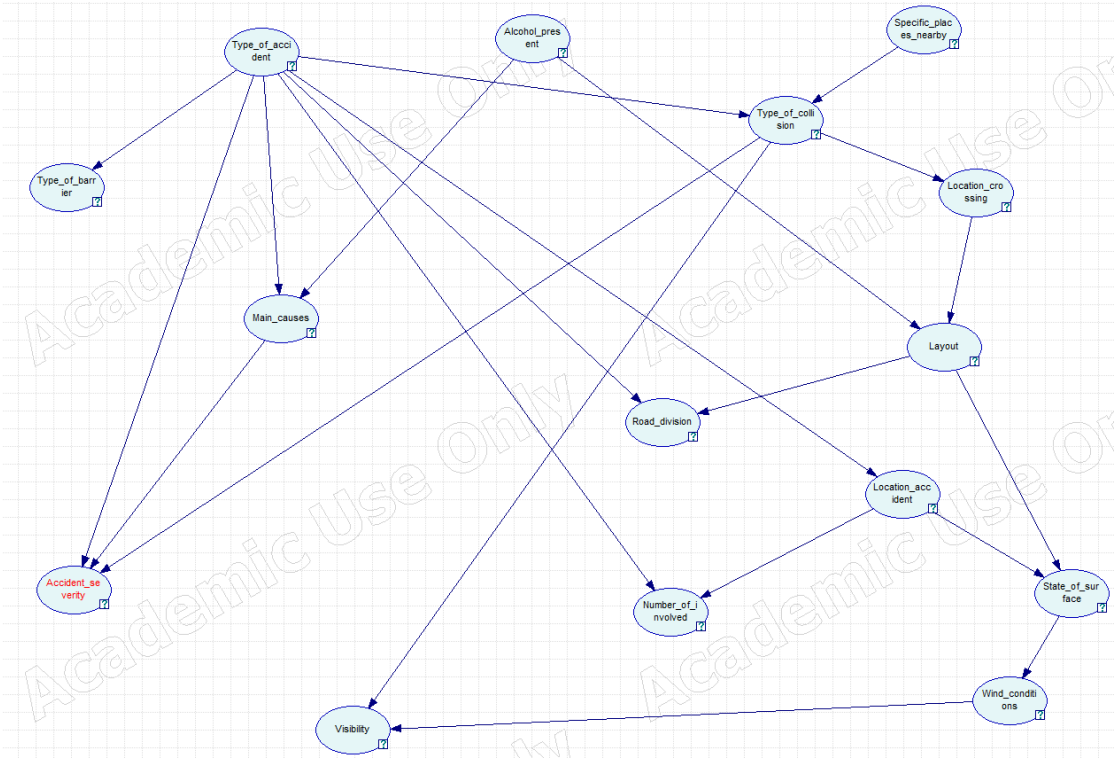

Figure 6.8: Final network

Accident Severity is chosen from the expert network. This means that the node will have three parents: type of accident, type of collision and main causes. This is a positive benefit that even very substantial information from the expert network is included in the merged one.

There are two other nodes selected from the expert network: Type of collision and Specific places nearby.

The node Location Accident or Location intersection would be normally selected from the expert network because of its better evaluation than in algorithm. Unfortunately, this is not possible, because at that moment a cycle would arise in the merged network.

We would also like to draw attention to the type of accident node, which cannot be selected from any input network (see more about this issue in section 5.5). The impossibility of this node selection from the input networks is also translated into the testing and validation score, see below.

Let us now find out if the merged network is better for data analysis than the input ones. Let us start with testing.

6.5.1 Testing results

We follow exactly the procedure in section 5.7. So first we perform testing, i.e. we calculate the scores of all nodes in the input and merged network based on the training data. We will use the prediction error as a method for calculating the score (section 2.6.3 and 5.7.2) and set the number of training data to one third of all records.

The numerical results are shown in Table 6.1. The most important information is in the last line. There can be seen that **the merged network has a better overall score than the input networks**.

Let us now focus on type of accident node, which we could not select from any network. We see exactly the situation we talked about when we had problems with cycles 5.5. The merged network has a worse score in this node than the input networks. In this case, we do not have to mind, because the resulting network still has the better overall score.

For other nodes, we see that the score of the node in the resulting network is always equal to the score of one of the input networks.

Node name	Expert net evaluaton	Algorithm net evaluation	Final net evaluation
Alcohol_present	590,41	590,41	590,41
State_of_surface	321,22	316,06	316,06
Wind_conditions	248,05	144,85	144,85
Visibility	806,83	794,36	794,36
Main_causes	614,09	511,32	511,32
Road_division	1028,11	608,61	608,61
Location_intersection	603,12	865,45	865,45
Type_of_accident	548,18	336,88	686,72
Location_accident	318,74	336,61	336,61
Specific_places_nearby	906,52	904,35	906,52
Type_of_collision	314,13	894,25	314,13
Type_of_barrier	493,08	2,00	2,00
Number_of_involved	458,58	254,82	254,82
Layout	862,92	114,78	114,78
Accident_severity	428,93	457,95	428,93
Overall	**8542,93**	**7132,95**	**6875,47**

Table 6.1: Evaluation scores of the expert, algorithm and the merged network.

6.5.2 Validation results

We will now move on to validation. Its course is exactly the same as in testing, we just use the rest of the data. This data has not been used at all so far.

The numerical results of the validation are given in Table 6.2. In the last line it holds that **the merged network has a better score again**.

Node name	Expert net evaluaton	Algorithm net evaluation	Merged net evaluation
Alcohol_present	172,40	172,40	172,40
State_of_surface	150,51	149,07	149,07
Wind_conditions	102,80	98,81	98,81
Visibility	327,23	323,21	323,21
Main_causes	190,01	167,98	167,98
Road_division	307,83	185,86	185,86
Location_intersection	178,38	263,52	263,42
Type_of_accident	165,99	106,43	209,46
Location_accident	79,98	91,14	91,20
Specific_places_nearby	263,47	261,39	263,44
Type_of_collision	94,87	268,61	94,87
Type_of_barrier	157,57	0,61	0,61
Number_of_involved	144,75	79,63	79,63
Layout	258,40	35,19	35,19
Accident_severity	137,77	136,56	137,77
Overall	**2731,96**	**2340,42**	**2272,93**

Table 6.2: Validation scores of the expert, algorithm and final network.

We have proved that the merged network (i.e. output of the algorithm) is better than the input networks and thus more suitable for doing data analysis, prediction, etc.

Conclusion

The research is focused on the algorithm for merging two Bayesian networks, into the final network, which will be better for data analysis in terms of quality than the original networks. This main goal can be considered fulfilled. Other goals were to describe this algorithm mathematically and to show it on the example of real data.

The algorithm is described in two ways: generally, where its individual steps are shown on simple graphical examples, and mathematically, where more complex notation is used. The algorithm was also utilized to demonstrate real data in the field of traffic accident analysis.

The main benefit of this algorithm is its versatility. This means that it can be used for any type of problem where its solver uses Bayesian networks.

The algorithm works on the principle of selecting edges from two input Bayesian networks containing the same nodes but different edges. It was primarily designed for merging networks suggested by an expert and an algorithm from data.

The procedure of selection consists in evaluating the nodes in the input networks by calculating their score based on given data. These selected nodes are then inserted into the final network with their input edges. Unfortunately, there may be cases where the above procedure cannot be used exactly.

This brings us to some open issues related to the algorithm for its further improvement:

- It is not possible to select a node from any input network due to Bayesian network definition (cycles). This is a problem because a better evaluation of the final network cannot be guaranteed.
- Conditional probability tables assigned to individual nodes in case they cannot be taken exactly from the input Bayesian networks.

These issues and suggestions for their solution are discussed in the relevant chapters of this work.

In addition, the algorithm can be extended in other directions. For example, a researcher may be convinced that an edge in one of the input networks must be preserved. This actually means that the final network should contain this edge regardless of the node score. This is not addressed at this time, as the final network would most likely have a worse rating.

This work demonstrates the use of an algorithm for the analysis of real traffic accident data. This is certainly not its only use. Another example may be a certain field in medicine, e.g. diabetology. where a doctor has data on patients and also has some idea of this field. Therefore, he can design an expert network based on his knowledge and independently generate a network from the data. Using an algorithm, he can then merge them and conduct data analysis using a better network. In each area of use of Bayesian networks, it is also possible to use a merging algorithm to refine the results of data analysis.

The author believes that the merging algorithm can be a very useful tool for researchers working with Bayesian networks not only today but also in the future. The author thinks that he has met all the goals mentioned in the introduction of this work and will continue to work on improving the merging algorithm to achieve even better merged networks.

Bibliography

[1] STEPHENSON, Todd Andrew. *An Introduction to Bayesian Network Theory and Usage.* IDIAP, 2000.

[2] DIESTEL, Reinhard. *Graph theory.* Fifth edition. Berlin: Springer, [2017]. Graduate Texts in Mathematics. ISBN 978-3-662-53621-6.

[3] NAGY, Ivan. *Lectures on Probability and Statistics [online].* Praha, 2020 [cit. 2020-11-08]. Available at: https://www.fd.cvut.cz/personal/nagyivan/Statistics/PrStLec20distant.pdf

[4] KÆRULFF, Uffe B.; MADSEN, Anders L. *Probabilistic Networks – An Introduction to Bayesian Networks and Influence Diagrams.* Aalborg University, 2005, 10-31.

[5] NAGY, Ivan. *Pokročilé Statistické Metody a Jejich Aplikace [online].* Praha, 2014 [cit. 2020-11-06]. Available at: https://www.fd.cvut.cz/personal/nagyivan/Doktorandi/LecturesPhD.pdf

[6] HECKERMAN, David. A Tutorial on Learning With Bayesian Networks. *Innovations in Bayesian networks*, 2008, 33-82.

[7] NEAPOLITAN, Richard E., et al. *Learning Bayesian Networks.* Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[8] *GeNIe Modeler* [online]. University of Pittsburgh, 2017 [cit. 2018-04-08]. Available at: https://www.bayesfusion.com/genie/

[9] *SMILE Wrappers* [online]. University of Pittsburgh, 2017 [cit. 2019-04-08]. Available at: https://support.bayesfusion.com/docs/Wrappers/

[10] D'AMBROSIO, Bruce. Inference in Bayesian Networks. *AI magazine*, 1999, 20.2: 21-21.

[11] MARGARITIS, Dimitris. *Learning Bayesian Network Model Structure from Data.* Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003.

[12] COOPER, Gregory F.; HERSKOVITS, Edward. A Bayesian Method for The Induction of Probabilistic Networks from Data. *Machine Learning*, 1992, 9.4: 309-347.

[13] NAGY, Ivan. *Odhad* [online]. Praha, 2017 [cit. 2020-11-20]. Available at: https://www.fd.cvut.cz/personal/nagyivan/WebLab/W_all.pdf

[14] DEL SAGRADO, Jos; MORAL, Serafin. Qualitative Combination of Bayesian networks. *International Journal of Intelligent Systems*, 2003, 18.2: 237-249.

[15] FENG, Guang; ZHANG, Jia-Dong; LIAO, Stephen Shaoyi. A Novel Method for Combining Bayesian Networks, Theoretical Analysis, And Its Applications. *Pattern Recognition*, 2014, 47.5: 2057-2069.

[16] JIANG, Chang-an; LEONG, Tze-Yun; KIM-LENG, P. O. H. PGMC: A Framework for Probabilistic Graphical Model Combination. In: *AMIA Annual Symposium Proceedings*. American Medical Informatics Association, 2005. p. 370.

[17] LI, Weihua; LIU, Weiyi; YUE, Kun. Recovering The Global Structure from Multiple Local Bayesian Networks. *International Journal on Artificial Intelligence Tools*, 2008, 17.06: 1067-1088.

[18] TARJAN, Robert. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1972, 1.2: 146-160.

[19] LAURITZEN, Steffen L. The EM Algorithm for Graphical Association Models with Missing Data. *Computational Statistics & Data Analysis*, 1995, 19.2: 191-201.

Appendices

Appendix A

Traffic Accident Data

The used traffic accident data were gathered from town council in year 2012. The data must be pre-processed before their usage. The variables, which are not feasible for this type of analysis or included in more than 90% the same value, are omitted. The data contains 3894 records.

The variable's description includes its name and all the values.

1. Type of an accident
   - an accident with non-tracked vehicle
   - an accident with parked or weaned vehicle
   - an accident with a solid barrier
   - other
2. Specific places and objects in a traffic accident area
   - crosswalk
   - other places (e.g. railway crossing, bridge, tunnel)
   - no specific place in a traffic accident area
3. Road division
   - two lanes
   - more than two lanes
   - none of presented options
4. Alcohol present
   - Yes
   - No
   - Not investigated

5. Location of an accident
    - lane
    - tramway tracks
    - none of presented options
6. Type of a collision of moving vehicles
    - front
    - side
    - back
    - not a collision of moving vehicles
7. State of a surface
    - dry
    - wet
    - snow
    - other (e.g. oil, mud)
8. Type of a solid barrier
    - not a solid barrier
    - solid barrier
9. Wind conditions
    - not complicated
    - snow or rain
    - other complicated
10. Location of an accident (other division)
    - out of a crossing
    - in a crossing (option 1)
    - in a crossing (option 2)
11. Layout
    - direct
    - curve
    - crossing
12. Main causes of an accident

- inadequate speed
- do not give priority
- bad driving style
- other

13. Visibility
    - day, good visibility
    - day, bad visibility
    - night
14. Number of involved vehicles
    - 1
    - 2
    - 3 or more
15. Accident severity
    - light
    - medium
    - serious

List of publications

- ŠTECHOVÁ, K., et al. Test For Insulin Pump Users - The Tool To Tailor Education And Check Patient's Knowledge. *Diabetes Technology and Therapeutics.* 2018, **20** A130. ISSN 1520-9156.
- ŠTECHOVÁ, K., et al. Sexual Dysfunction in Women Treated for Type 1 Diabetes and the Impact of Coexisting Thyroid Disease. *Sexual Medicine.* 2019, **7**(2), 217-226. ISSN 2050-1161. DOI 10.1016/j.esxm.2019.03.001.
- LOKAJ, Z., et al. Technical part of evaluation solution for cooperative vehicles within C-ROADS CZ project. In: RŮŽIČKA, J., ed. *2020 Smart City Symposium Prague.* Prague, 2020-06-25. New York: IEEE Press, 2020. ISBN 978-1-7281-6821-0. DOI 10.1109/SCSP49987.2020.9133885.
- TABERY, K., et al. Continuous glucose monitoring as a screening tool for neonatal hypoglycemia in infants of diabetic mothers. *The Journal of Maternal-Fetal & Neonatal Medicine.* 2020, 1889-1894. ISSN 1476-7058. DOI 10.1080/14767058.-2018.1533941.
- ŠTECHOVÁ, K., et al. Lessons Learned from Implementing a New Testing/Educational Tool for Patients Using an Insulin Pump. Diabetes Technology and Therapeutics. 2018, 20(8), 524-530. ISSN 1520-9156. DOI 10.1089/dia.2018.0095.
- VANIŠ, M. and I. NAGY. Merging Bayesian Networks based on different types of input knowledge. In: NOUZOVSKÝ, L., et al., eds. *Young Transportation Engineers Conference 2018.* Praha, Horská 3, 2018-11-01. Praha: CTU. Faculty of Transportation Sciences, 2018. p. 53-61. ISBN 978-80-01-06464-1.
- VANIŠ, M. and K. URBANIEC. Employing Bayesian Networks and Conditional Probability Functions for Determining Dependences in Road Traffic Accidents Data. In: RŮŽIČKA, J., ed. *2017 Smart Cities Symposium Prague (SCSP) - IEEE PROCEEDINGS.* 2017 Smart Cities Symposium Prague (SCSP), Prague, 2017-05-25/2017-05-26. New York: IEEE Press, 2017. ISBN 978-1-5386-3825-5.
- PŘIKRYL, J. and M. VANIŠ. Comparing numerical integration schemes for a car-following model with real-world data. In: CHLEBOUN, J., et al., eds. *Proceedings of Seminar Programs and Algorithms of Numerical Mathematics 18.* Programy a

algoritmy numerické matematiky 18, Janov nad Nisou, 2016-06-19/2016-06-24. Praha: Matematický ústav AV ČR, v. v. i., 2017. p. 89-96. ISBN 978-80-85823-67-7. DOI 10.21136/panm.2016.11.

Curriculum Vitae

Education

2013 – 2015	**CTU in Prague Faculty of Transportation Sciences**, *Master's degree programme:* Engineering informatics in transport and communications *Diploma thesis:* Verification of microscopic traffic flow models on real traffic data.
2009 – 2013	**CTU in Prague Faculty of Transportation Sciences**, *Bachelor's degree programme:* Automation and informatics *Bachelor thesis:* Modelling Selected Problems in Transportation in Java.
2005 – 2009	**Gymnasium of Professor Jan Patočka**, Secondary education

Work experience

Projects within CTU in Prague Faculty of Transportation Sciences

2017 – 2021	Project C-Roads – The platform of harmonised C-ITS deployment in Europe Member of WG3 - Evaluation & Assessment
2021 – 2025	TAČR - Electronic speed limitation of vehicles in emergency and crisis situations triggered by security forces
2021 – 2022	TAČR - Protection of non-personal data and databases in autonomous systems
2020 – 2022	TAČR - Research of alternative methods of position determination and its integrity with GNSS for drivers using C-ITS
2020 – 2022	TAČR - Privacy and personal data proteciton in autonomous driving systems

2021	Student Grant Competition of CTU Extension of independent mixtures method for discrete data analysis

Finished projects within CTU in Prague Faculty of Transportation Sciences

2018	Student Grant Competition of CTU Synergetic synthesis of Bayesian networks
2019	Student Grant Competition of CTU Utilization of non-categorical distributions in discrete data analysis
2020	Student Grant Competition of CTU Prediction of discrete data using model of independent mixtures
2016 – 2018	Project MAVEN – Managing Automated Vehicles Enhances Network
2014 – 2015	TAČR - The extended data model for the disable people and the methodology of its interpretation in the navigation

Other

2015 – now	**Motol University Hospital** Department of Internal Medicine, University Hospital Motol and 2nd Medical Faculty of Charles University Statistical analysis, Application programming
2020 – now	**Prague 1 District office** Transport consultant
2014 – 2016	**CTU in Prague** The member of the CTU Academic Senate