CS5011 P3

160004864

18 November 2019

1 Introduction

For this assignment we were required to design and document two variants of a Bayesian network solution for predicting the risk of traffic congestion at the entrance of Meridonia City Hospital[1] and develop an agent that is able to construct a Bayesian network and make probabilistic inferences using the variable elimination algorithm.

1.1 Implementation Checklist

1. Constructed BN1, a Bayesian network representing the problem in the specification under the assumption that the factors mentioned are independent.
2. Constructed BN2, a Bayesian network representing the problem in the specification, with relaxed independence assumptions.
3. Explained how the above designs differ and how the probabilities of the network's nodes were assigned.
4. Designed three queries, one predictive one diagnostic and a profiling one, that can be answered by both networks. I've also included a discussion on the differences between the answers of the two networks.
5. Implemented a Bayesian inference agent that is able to construct a Java representation of a Bayesian network and then use the variable elimination algorithm to make predictive queries.

1.2 Compiling and Running Instructions

In order to run the program implementing the Bayesian inference agent, first navigate to the base directory, **CS5011_P3**. Then, in order to run the program using the already compiled source code, use the following command:

```
java -cp out/production/CS5011_P3 A3main
```

In order to re-compile the source code, you may use the following command:

```
javac -d out/production/CS5011_P3 A3src/*.java
```

2 Design, Implementation and Evaluation

2.1 Part 1

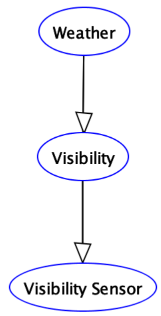
2.1.1 Domain Variables

The first step in constructing the required Bayesian Networks was to identify the different variables in the problem domain:

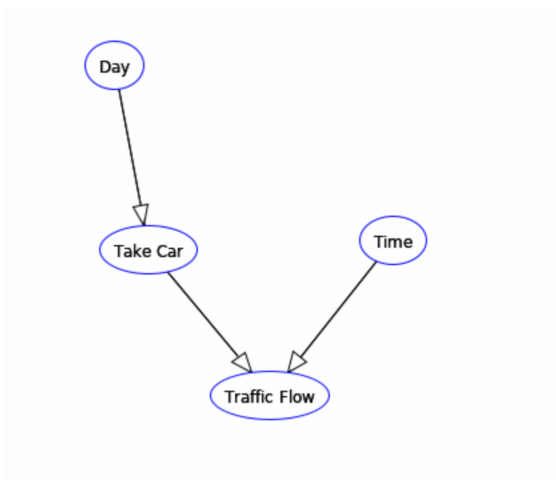
- Weather - this represent the weather conditions of Meridonia. Its domain is `sunny, overcast, raining, snowing`.
- Visibility - this represents the visibility conditions in Meridonia. Its domain is `high, low`.
- Visibility Sensor - this represents the sensor which is used to detect visibility parameters. Its domain is `high, low`.
- Day - this represents whether the day is a holiday or a working day. Its domain is `working, holiday`.
- Take Car - this represents the probability of people taking their car to circulate around the area. Its domain is `true, false`.
- Time - this represents whether the time is peak time or not. Its domain is `peak, off`.
- Traffic Flow - this represents the 'flow of the traffic'. In essence this is the volume of traffic. Its domain is `high, low`.
- Camera - this represents the camera in proximity of the hospital entrance to detect the traffic flow. Its domain is `high, low`.
- Traffic Congestion - this represents traffic congestion. It's domain is `true , false`.
- Alarm - this represents the alarm that alerts the emergency services in the event of traffic congestion being predicted. Its domain is `true, false`.

2.1.2 Relationships

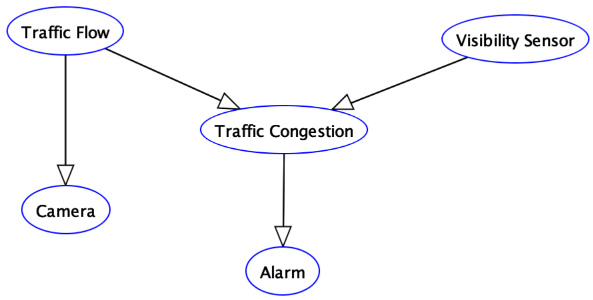
The next step in constructing the Bayesian networks was to identify the relationships between the domain variables. We shall start with the relationships identified under the assumption that the factors are independent (BN1).



The first relationship identified is that the weather will affect the visibility which in turn affects the reading of the visibility sensor.



This next set of relationships show that the type of day has an effect on the probability of people taking their cars and the probability of people taking their cars has an effect on the traffic flow. Furthermore, the time of day is significant as the traffic flow will increase during peak times, hence time has an effect on traffic flow.



In this part of the network there are several relationships to be considered. Firstly, we can see the prediction of traffic congestion is affected by the traffic flow and the visibility sensor reading. The traffic congestion probability has an effect on the probability that the alert system will be activated. The camera is not used as a variable for predicting traffic congestion as the specification states its purpose is to indicate whether there is high or low traffic flow and not for triggering alerts. The complete, resulting network is as follows:

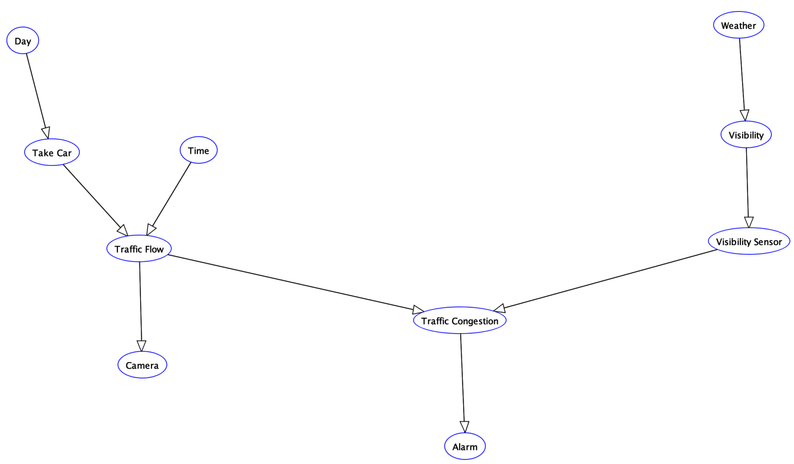


Figure 1: BN1, saved as bn1.xml

Now that we have constructed BN1, we shall construct another Bayesian network, BN2, with relaxed independence assumptions.

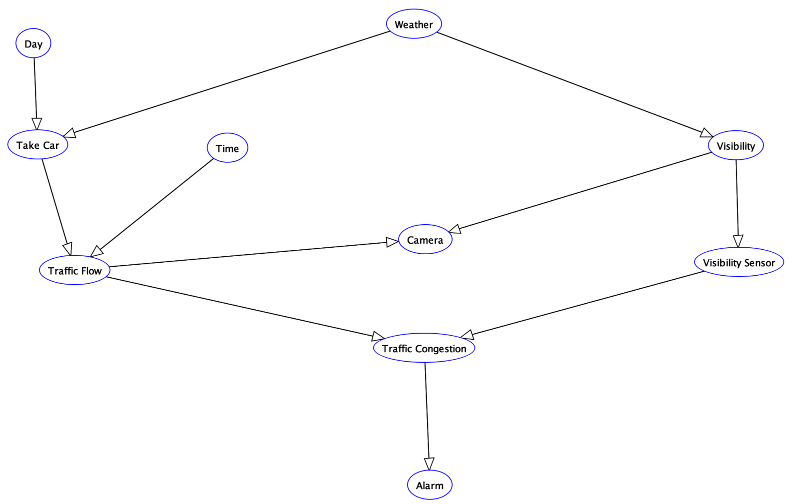


Figure 2: BN2, saved as bn2.xml

As we can see the following additional relationships have been added:

- The weather affects the probability of people taking their cars. Intuitively this is because if for example it is snowing, Meridonians are much more likely to choose to work remotely or take the day off. On the contrary if it's a sunny holiday, Meridonians will be more likely to get their car and go out with their families.
- The visibility affects the probability of the camera giving an accurate reading. For example, if visibility is low, the camera may not be able to accurately distinguish what the traffic flow is like.

2.1.3 Probability tables

The next step in constructing our Bayesian networks was the assignment of prior and posterior probabilities. In this section I will be explaining how the probabilities were assigned.

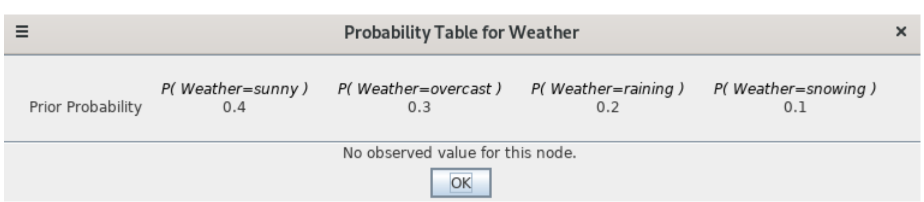


Figure 3: Weather probability table.

I assigned these probabilities based on Meridonia's climate. Its climate is a continental one[1] with a warm summer, moderately wet winter and potentially severe snowfall during winter.

Probability Table for Visibility

Weather	P(Visibility=high)	P(Visibility=low)
sunny	1.0	0.0
overcast	0.7	0.3
raining	0.3	0.7
snowing	0.0	1.0

No observed value for this node.

OK

Figure 4: Visibility probability table.

I assigned these probabilities based on a few assumptions. Firstly, I assumed that visibility is always high when it is sunny, and always low when it is snowing. Secondly, I assumed that during overcast there is a 0.7 probability that there will be high visibility and when it is raining that probability will be lower, at 0.3.

Probability Table for Visibility Sensor

Visibility	P(Visibility Sensor=high)	P(Visibility Sensor=low)
high	0.95	0.05
low	0.05	0.95

No observed value for this node.

OK

Figure 5: Visibility probability table.

I assigned these probabilities based on the system description in the specification. It states that the sensor gives the wrong value with 5% probability. Hence it will be correct 95% of the time i.e. report a high visibility when visibility is high and report low visibility when the visibility is low.

Probability Table for Day

	P(Day=working)	P(Day=holiday)
Prior Probability	0.78	0.22

No observed value for this node.

OK

Figure 6: Day probability table.

I assigned these probabilities based on the information that Meridonia has 80 holidays in the 360 days. Thus there is a 0.22 probability that each given day will be a holiday.

Probability Table for Take Car

Day	P(Take Car=true)	P(Take Car=false)
working	0.6	0.4
holiday	0.3	0.7

No observed value for this node.

OK

Figure 7: Probability table for people taking their car.

I assigned these probabilities based on the national statistics which show that on holiday periods the probability of people taking their cars is at 30% and on working days that figure is 60%.

Probability Table for Time

	P(Time=peak)	P(Time=off)
Prior Probability	0.375	0.625

No observed value for this node.

OK

Figure 8: Time probability table.

I assigned these based on the hospital statistics showing that the peak time for accessing the hospital is between 6-10am and 5-10pm. That is 9 hours in a 24 hour day, and thus any given hour is a peak time with 0.375 probability.

Probability Table for Traffic Flow

Time	Take Car	P(Traffic Flow=high)	P(Traffic Flow=low)
peak	true	1.0	0.0
peak	false	0.4	0.6
off	true	0.6	0.4
off	false	0.0	1.0

No observed value for this node.

OK

Figure 9: Traffic flow probability table.

I assigned these probabilities based on a series of assumptions. Firstly, I assumed that during a peak time in a day in which people take their cars, the probability of the traffic flow being high is 1.0, whereas if people don't take their cars, the probability for high traffic flow goes down to 0.4. However, during a non-peak time traffic flow will be high if people take their cars with a probability of 0.6 and 0 if they don't. I assumed that the people taking their cars is a more significant contributor to high traffic flow, hence the non-symmetrical probability table.

Probability Table for Camera

Traffic Flow	P(Camera=high)	P(Camera=low)
high	0.95	0.05
low	0.05	0.95

No observed value for this node.

OK

Figure 10: Camera probability table.

I assigned these probabilities based on the information in the specification stating that the camera has a bad fault tolerance and gives the wrong value with 5% probability. Hence it will only report the correct value i.e. that there is a high traffic flow when it is high and report that traffic flow is low when it is low, with a 0.95 probability.

Probability Table for Traffic Congestion

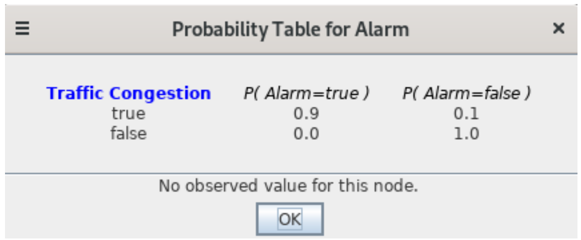
Visibility Sensor	Traffic Flow	P(Traffic Congestion=true)	P(Traffic Congestion=false)
high	high	0.65	0.35
high	low	0.05	0.95
low	high	0.95	0.05
low	low	0.25	0.75

No observed value for this node.

OK

Figure 11: Traffic congestion probability table.

I assigned these probabilities based on a series of assumptions. I firstly assumed that traffic flow is the main contributing factor to traffic congestion. As a result I assigned the probability of traffic congestion being true at 0.65 when traffic flow and visibility is high, whereas the probability of traffic congestion begin true when traffic flow is low and visibility is also low, is 0.25. Then I assigned the probability of traffic congestion to be true when traffic flow is high under low visibility conditions to be 0.95 and 0.05 when traffic flow is low under high visibility conditions.



Probability Table for Alarm

Traffic Congestion	P(Alarm=true)	P(Alarm=false)
true	0.9	0.1
false	0.0	1.0

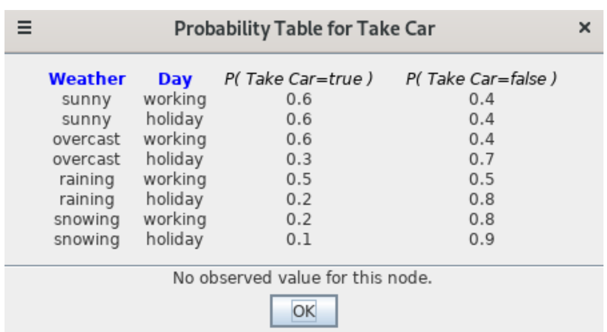
No observed value for this node.

OK

Figure 12: Alarm probability table.

I assigned these probabilities using the information given to us in the specification with a minor assumption. Since the alarm is triggered only 90% of the times to avoid false alarms, the alarm will not not trigger when traffic congestion is predicted to be true 10% of the time. However I assumed that the opposite does not occur i.e. the alarm is not triggered 10% of the time when traffic congestion is not predicted. Hence the alarm will never trigger if the traffic congestion prediction is false.

In section 2.1.2, we introduced two additional relationships for the construction of BN2, namely that the weather affects whether the people take their car, and visibility affects the probability of the camera reporting the correct value. As a result, the probability tables for 'Take Car' and 'Camera' will change.



Probability Table for Take Car

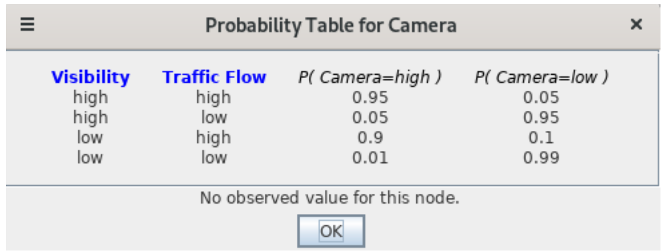
Weather	Day	P(Take Car=true)	P(Take Car=false)
sunny	working	0.6	0.4
sunny	holiday	0.6	0.4
overcast	working	0.6	0.4
overcast	holiday	0.3	0.7
raining	working	0.5	0.5
raining	holiday	0.2	0.8
snowing	working	0.2	0.8
snowing	holiday	0.1	0.9

No observed value for this node.

OK

Figure 13: Take Car probability table for BN2.

In the case where the day is a holiday, people are more likely to take their car if it is sunny as they are more likely to want to go out with their friends and family and enjoy the weather on their day off. The probability of people taking their car on their holiday becomes lower as the weather becomes worse. On the contrary if it's a working day, the probability of people taking their car is the same regardless whether the weather is sunny or if there's an overcast. However, this probability becomes lower if it's raining as people may find driving in the rain difficult and thus they're more likely to work from home. This effect is more pronounced when it's snowing as there may even be road closures and thus people are less likely to take their cars, even if it's a working day.



Probability Table for Camera

Visibility	Traffic Flow	P(Camera=high)	P(Camera=low)
high	high	0.95	0.05
high	low	0.05	0.95
low	high	0.9	0.1
low	low	0.01	0.99

No observed value for this node.

OK

Figure 14: Camera probability table for BN2.

The probabilities of the camera reporting the correct value when visibility is high is the same as in BN1, however when there's low visibility, the camera is more likely to report the wrong value. Specifically the camera is twice as likely to report the wrong value, compared to BN1.

The following are the two networks including all of the assigned and derived probabilities.

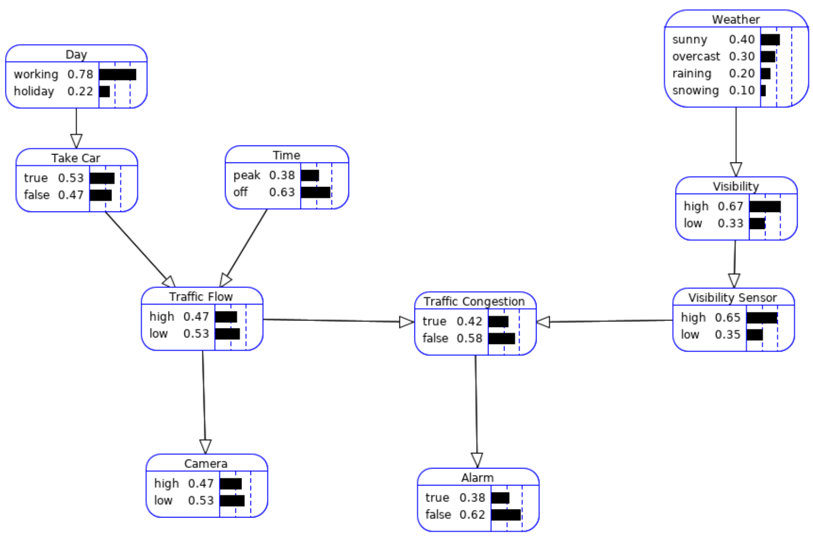


Figure 15: Probabilities for BN1.

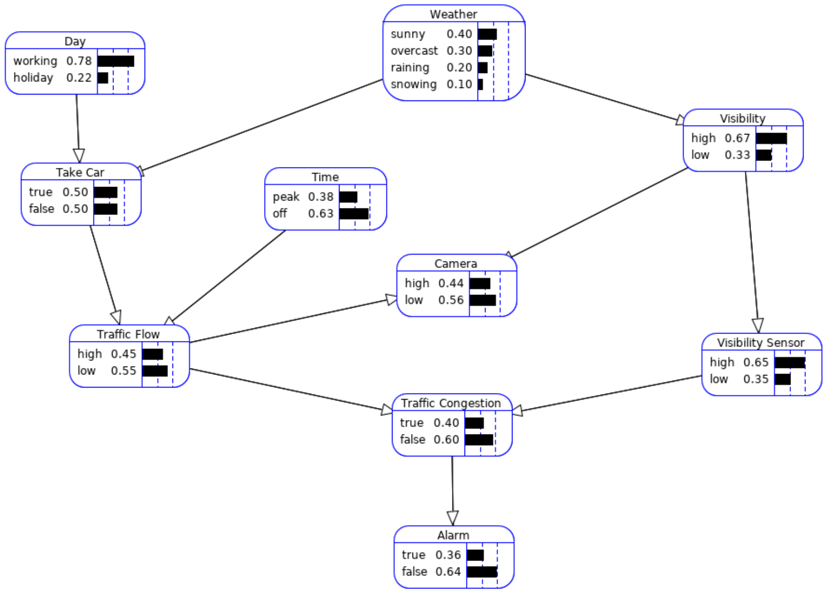


Figure 16: Probabilities for BN2.

2.1.4 Queries

In this section we will design three queries, a predictive, a diagnostic and a profiling query and compare the answers of the two networks.

- Predictive query: We observe that the visibility sensor reports that visibility is low. How likely is is that the alarm triggers? Since the value reported by the visibility sensor is used to predict traffic congestion which in turn affects whether the alarm is triggered or not, it is expected that the probability of the alarm triggering increases from the one shown in figures 15 and 16.

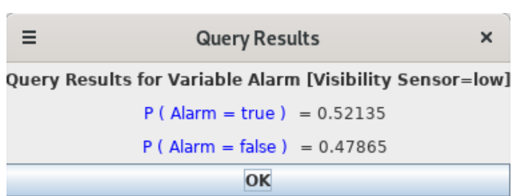


Figure 17: BN1's answer to predictive query.

As shown by the screenshot, BN1's answer to the query is 0.52135.

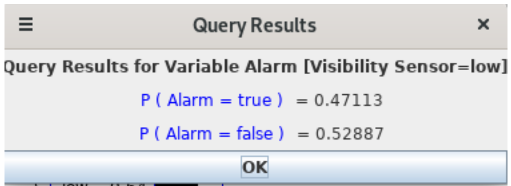


Figure 18: BN2's answer to predictive query.

As shown by the screenshot above, BN2's answer to the query is 0.47113.

- Diagnostic query: We observe that traffic congestion is true. How likely is it for the day to be a holiday? Since people are more likely to take their car during a working day, increasing traffic flow which in turn contributed to congestion, we expect that the probability of the day being a holiday decreases from the one shown in figures 15 and 16.

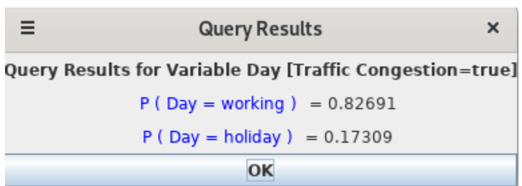


Figure 19: BN1's answer to diagnostic query.

As shown by the screenshot above, BN1's answer to the query is 0.17309.

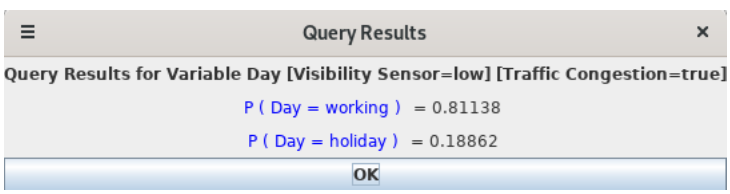


Figure 20: BN2's answer to diagnostic query.

As shown by the screenshot above, BN2's answer to the query is 0.18862.

- Profiling query: How does the Bayesian network solution react depending on the visibility is high? Before looking at how each variable might be affected let's consider which variables will be affected by the observation in each of the two networks i.e. which variables are not conditionally independent to 'Visibility'. In BN1, the variables that will be affected are 'Weather', 'Visibility Sensor', 'Traffic Congestion' and 'Alarm'. When

'Visibility' is observed to be high, I estimate that the probability for 'Traffic Congestion' to decrease, 'Visibility Sensor' will be more likely to report 'high', 'Alarm' is more likely to be false (not trigger) and the 'Weather' will be more likely to be sunny.

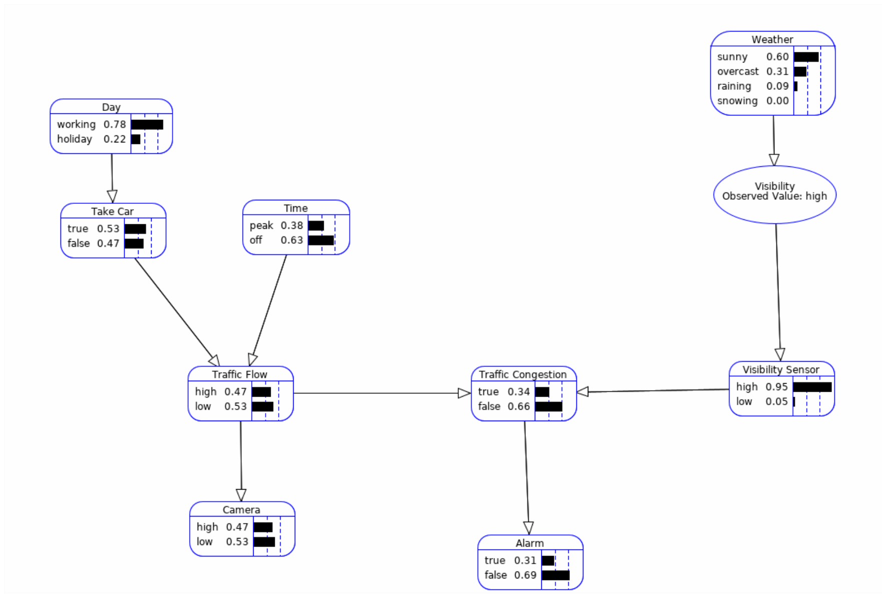


Figure 21: BN1's answer to profiling query.

Comparing the above figure to figure 15, we can see that the estimations mentioned above were correct.

In the case of BN2, I estimate that in addition to the variables affected in BN1, given the new relationships introduced, the probability of 'Take Car', 'Traffic Flow' and 'Camera' will be affected. Furthermore, when visibility is set to high, I estimate that the probability of 'Take Car' will increase since people are more likely to take their car if it's a holiday, and thus the probability of 'Traffic Flow' being high is expected to increase as well. Furthermore, I expect that the probability of the 'Camera' reporting high traffic flow will also increase when 'Visibility' has been observed to be high.

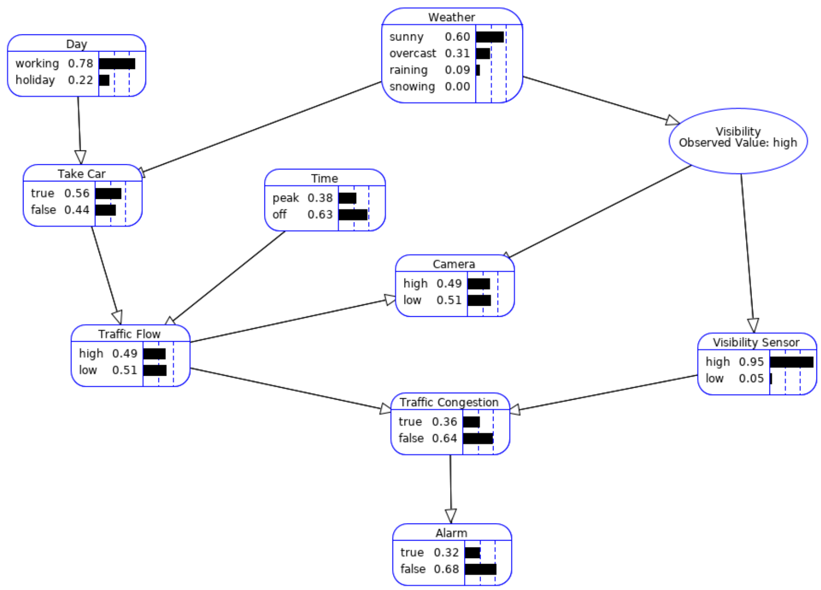


Figure 22: BN2's answer to profiling query.

Comparing the above figure to figure 16, we can see that the estimations mentioned above were correct.

2.2 Part 2

For this part of the assignment we were required to develop an agent that can construct a Bayesian network and make inferences using the variable elimination algorithm as described in the lectures[2].

Note: Due to the word limit on this report, I don't go into great depths describing the implementation of the agent. However, the source code is heavily commented and the reader is encouraged to review it in order to gain a better understanding of how each feature is implemented.

2.2.1 Constructing the Bayesian network

The Bayesian network is represented in the program as a series of nodes, implemented in the Node.java file, with parent-child relationships between them. Each of these nodes have certain attributes such as a label, an array holding their probability table, a pointer to their parent node and a list of pointers to their children nodes. Furthermore, since the network is a singly connected network, there will only be one node without a parent (this is the root node) which is indicated by a `boolean` attribute. The network is constructed in the Network class, implemented in the Network.java file, which creates all the nodes, setting all of their attributes and creating the relationships between them. The network that is used to verify the correctness of the agent's implementation is the one found in the `BNpart2.xml` file.

2.2.2 Creating the query

A predictive query involves calculating the probability table of a random variable after an observation has been made i.e. we have evidence for the value of a certain random variable. Upon running the program the user is shown the list of random variables that can be queried. Once the user enters valid input, the program displays the list of random variables that can be used as evidence as part of the predictive query. These will be the ancestors of the variable that was chosen to be queried. Finally, the program prompts the user for the value of the observed variable and this will either be true or false.

2.2.3 Projecting the evidence

Once the query has been formed, the program will us the `applyEvidence` method which carries out the process of projecting the evidence using a process similar to marginalisation. Consider the probability table of the alarm variable in the fire alarm belief network (FBN):

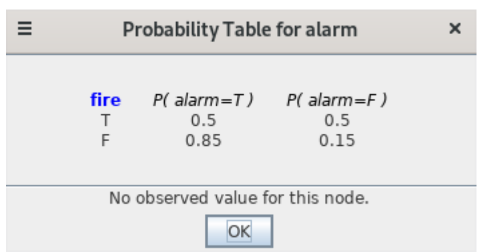


Figure 23: The alarm probability table in the FBN

If the variable fire is observed to be true, we can take the new probability table for alarm to consist only of `P(alarm=T)= 0.5` and `P(alarm=F)= 0.5`.

2.2.4 Variable elimination

Once the evidence has been projected, the `joinAndMarginalise` method will carry out the variable elimination algorithm. This method takes a list of `Node` objects as a parameter which represent the variables on which the join and marginalise process has to take place. The list consists of nodes from the variable that is being queried up to (but not including) the evidence variable. Starting from the child of the observed variable, the new probability tables are calculated on the fly and once the new probability table of the queried value is constructed, the program will display the results. An example of how these calculations are carried out can be found in the lecture notes[2].

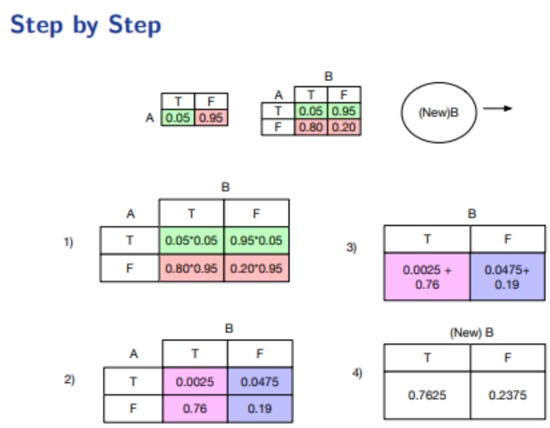


Figure 24: A step in the join and marginalise algorithm.

2.3 Evaluation

2.3.1 Part 1 Evaluation

Overall, I'm very happy with the designs I have created as I believe they reflect the scenario we were given very well. I was very meticulous in translating the scenario into nodes to be used in the Bayesian networks and I have used all the data available to us to fill out the probability tables. Furthermore, I made sure to justify all of my decisions when filling out the probability tables that we had no information for. I didn't find this part particularly difficult, although modelling the relationship between the different random variables required careful consideration.

2.3.2 Part 2 Evaluation

I'm very pleased with the inference agent I have designed as it's able to correctly construct a Bayesian network and perform predictive queries on it. What I found most difficult about this part applying the join and marginalise algorithm as it involved a series of steps that had to be carefully made to get the correct result. If I had more time I would first modify the program so that it can perform queries on networks in which nodes may have more than 1 parents and more than 2 children. Furthermore, I would also extend the program's functionality by being able to answer diagnostic queries in addition to the predictive queries. Lastly I would consider implementing a feature which enables the program to construct a Bayesian network by parsing an XML file. Despite not being part of the learning outcomes, this would've made loading and querying different Bayesian networks quicker.

3 Testing

In order to prove the correctness of the program created in Part 2, I have conducted a series of tests to see if the network gives the correct answers to the various predictive queries. For the intents and purposes of doing so I will be using the fire alarm belief network and compare the answers of my agent to the answers given by the AIspace tool.

- Query report when fire is observed to be true:

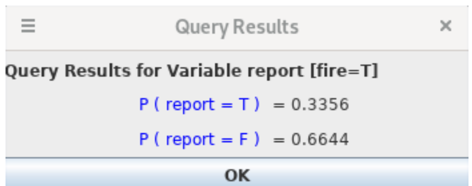


Figure 25: AIspace result of query for report when fire has been observed as true.

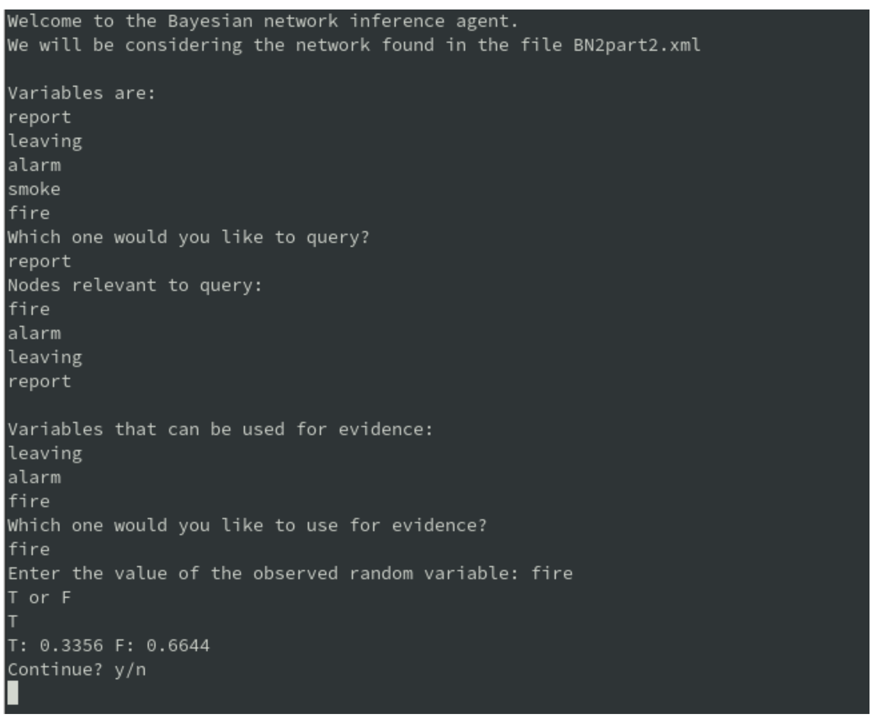


Figure 26: Inference agent result of query for report when fire has been observed as true.

- Query leaving when alarm is observed to be false:

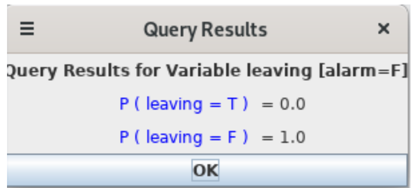
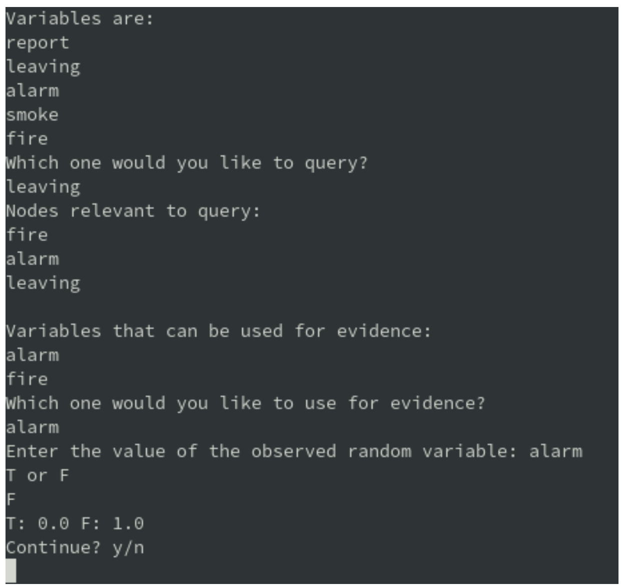


Figure 27: AIspace result of query for leaving when alarm has been observed as false.



```
Variables are:
report
leaving
alarm
smoke
fire
Which one would you like to query?
leaving
Nodes relevant to query:
fire
alarm
leaving

Variables that can be used for evidence:
alarm
fire
Which one would you like to use for evidence?
alarm
Enter the value of the observed random variable: alarm
T or F
F
T: 0.0 F: 1.0
Continue? y/n
```

Figure 28: Inference agent result of query for leaving when alarm has been observed as false.

- Query report when alarm is observed to be true:

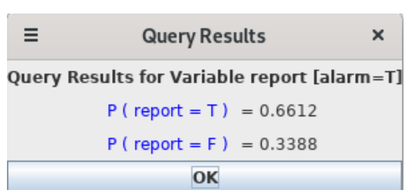
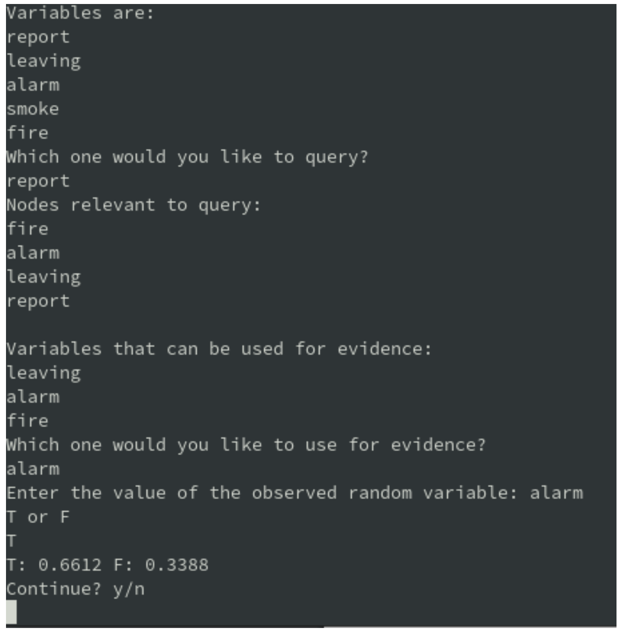


Figure 29: AIspace result of query for report when alarm has been observed as true.



```
Variables are:
report
leaving
alarm
smoke
fire
Which one would you like to query?
report
Nodes relevant to query:
fire
alarm
leaving
report

Variables that can be used for evidence:
leaving
alarm
fire
Which one would you like to use for evidence?
alarm
Enter the value of the observed random variable: alarm
T or F
T
T: 0.6612 F: 0.3388
Continue? y/n
```

Figure 30: Inference agent result of query for report when alarm has been observed as true.

As we can see from the examples above, the inference agent that I have implemented gives exactly the same answers to the predictive queries as the AIspace tool. Thus if we assume the correctness of the AIspace tool then we can deduce that the inference agent is also correct.

4 Conclusion

I am very pleased with the results of my efforts as I have fully completed the requirements of the specification. Namely, for Part 1 I have successfully designed a Bayesian network which predicts the risk of traffic congestion at the entrance of the fictional Meridonia City Hospital and I have also successfully implemented an agent which is able to create a Java representation of a Bayesian network and give answers to predictive queries, using the variable elimination algorithm. This assignment was not only very interesting, but it was also very successful in achieving its learning outcomes as I feel that it has improved my ability in understanding and using Bayesian networks and probabilistic inferences.

References

[1] https://wiki.opengeofiction.net/wiki/index.php/Meridonia

[2] Toniolo, A. (2019). CS5011(L14 W8): CS5011 Uncertainty - Part B