CS5011 A3 Report

matriculation number: 160021429

1. Introduction

The main aim of this practical is to gain understanding on how to model and use Bayesian Networks to reason with uncertainty. The part 1 asked us to design and construct the BN(Bayesian Network)s to represent the given problem. And the part 2 asked us to implement a variable elimination algorithm for predictive query with Java. For the extension, I improved my Java program to handle all acyclic networks with predictive query.

1 - 1. How to compile and run the program for part 2

To compile : “javac *.java”

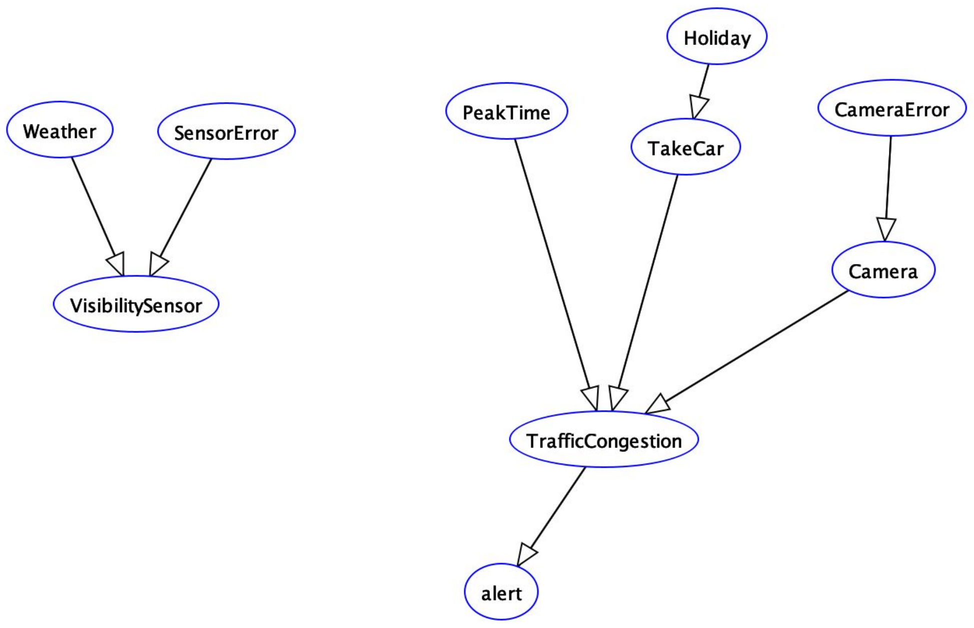
To run the program: either “java A3main” or “java A3main <file_path_of_xml_file>”

2. Design and Implementation

2 - 1. Part 1

For the part 1, we are asked to implement 2 BNs. The BN1 represents the given problem under the assumption that the factors are all independent. Therefore, the Bayesian Network in the bn1.xml should be a singly connected network. However, the BN2 can contain multiply connected network, since we do not need to care about the assumption that we made for bn1.xml.

2 - 1 - 1. BN1.xml



<bn1.xml>

The image above is the bayesian network that I designed to display all features that are listed in the specification.

The first feature is weather. The weather could be one of Sunny, Overcast, Raining and Snowing. Clearly, the weather could affect the value of visibility sensor, thus, I connected the Weather node and VisibilitySensor node.

The SensorError is a feature that represents the sensor error. According to the specification, the possibility of sensor error is 0.05, thus, P(SensorError = T) = 0.05 and P(SensorError=F)

= 0.95. I also connected the SensorError node to the VisibilitySensor node, since the SensorError node represents the possibility that the visibility sensor gives the wrong value.

The TrafficCongestion is a feature that represents the prediction of traffic congestion. The prediction of traffic congestion is the main part of the system. And there are many features that affect the prediction of traffic congestion - 1) PeakTime, 2) TakeCar, and 3) Camera.

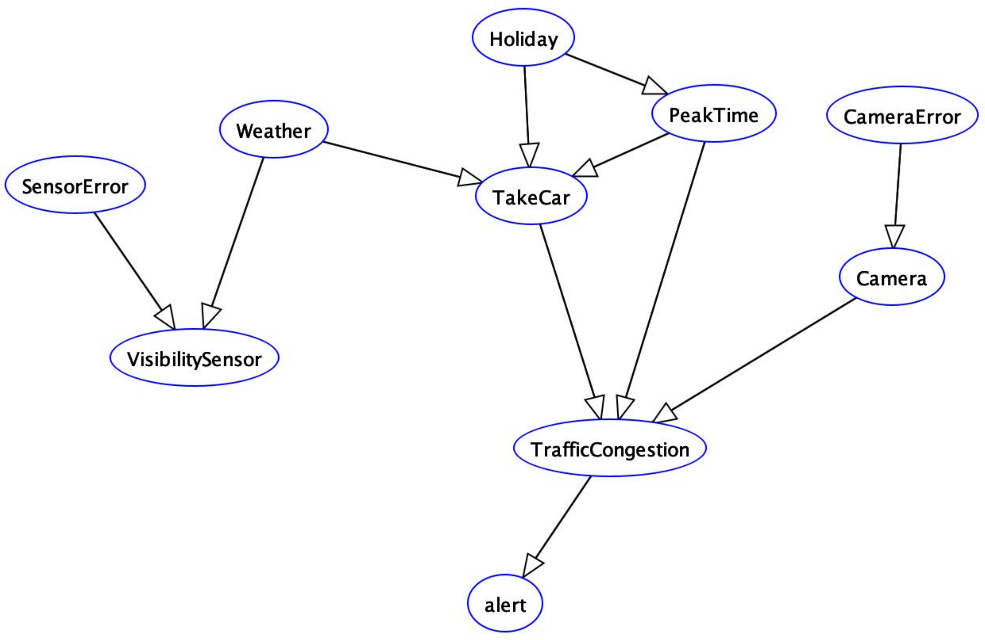
The PeakTime is the feature that represents if the current time is whether peak time or not. According to the spec, the peak time for accessing the hospital is between 6-10 am and 5-10pm. Thus, 9 / 24 = 0.375 is the probability of peak time (P(PeakTime = T) = 0.375).

The TakeCar is the feature that represents the probability of the visitors take cars. And the TakeCar is affected by the Holiday. The Holiday is the feature that represents whether it is a working day or a holiday day. The specification says that there are 80 days holidays in 360 days. Thus, the probability of holiday should be 80 / 360 = 0.22.

The Camera is the feature that shows whether there is high or low traffic flow. Similar to the visibility sensor, the camera could also give a wrong value. Therefore, I add a node called CameraError. The camera has a bad fault tolerance and may give the wrong value with 5% probability. So the value of P(CameraError = T) should be 0.05, and P(CameraError = F) should be 0.95.

Since the final goal of the alert system is to alert the traffic congestion, I added a node “alert” to the bayesian network. The alert system should alert when the prediction system predicts the traffic congestion. Thus, it is possible to say that the Alert feature is dependent on the TrafficCongestion. According to the specification, the alert is triggered only 90% of the times, to avoid false alarms. Henceforth, the value of P(alert = T | TrafficCongestion = T) should be 0.9, and the value of P(alert = T | TrafficCongestion = F) should be 0.1.

2 - 1 - 2. BN2.xml



<bn2.xml>

As you could see above, the structure of the bn2.xml is almost same with the bn1.xml. However, the bayesian network in the bn2.xml has more connections.

To add more connections, I made 2 assumptions. 1) The peak time of the general days is different with the peak time of the holiday. 2) The probability of people take their car, they might consider not only whether it is holiday or not but also if it is peak time now and the weather.

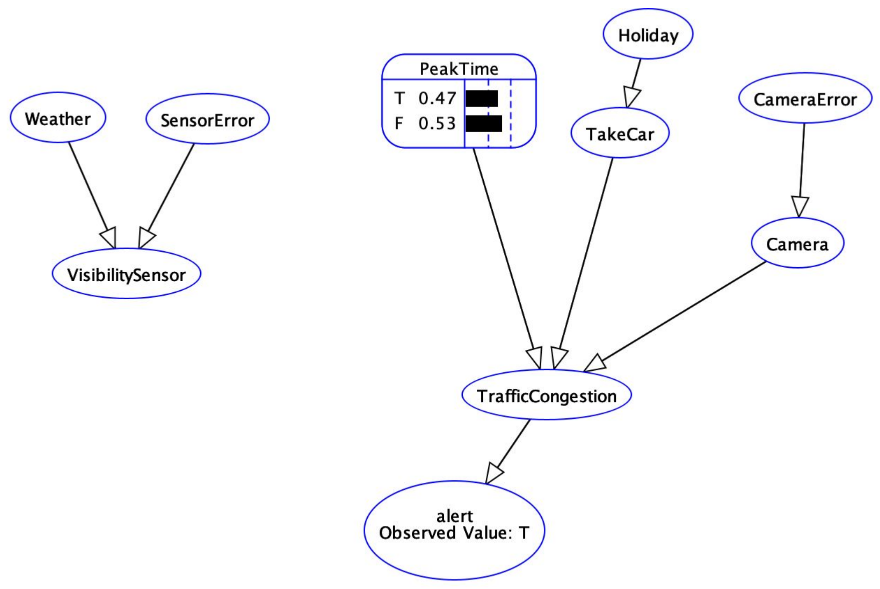
There is a possibility that the working time of the hospital changes on the holiday. Thus, the peak time might change on holidays. Furthermore, people might check not only if it is a holiday but also the weather and time. Henceforth, it is possible to say that the Holiday affects to the PeakTime feature, and both PeakTime and Weather affects to the probability of the TakeCar feature. These are the main reasons that I added 3 new connections to the bayesian network.

In the bn1.xml, the probability of the TakeCar feature only depends on the Holiday feature. However, now in the bn2.xml, the TakeCar feature is affected by 3 different features - Holiday, PeakTime, and Weather. Thus, now we need to calculate more complex expression to calculate the probability of the TakeCar. In the bn1.xml, we were able to get the probability by calculate P(TakeCar) = P(TakeCar | Holiday) * P(Holiday). However, now we need to use P(TakeCar) = P(TakeCar | Holiday, PeakTime, Weather) * P(Holiday) * P(PeakTime) * P(Weather).

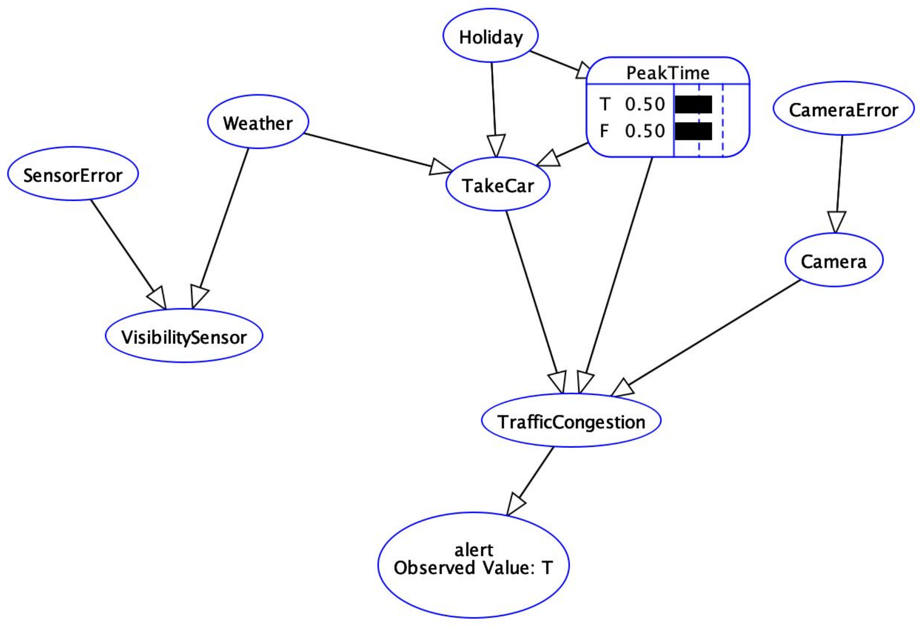
2 - 1 - 3. Queries

A. Diagnostic

With the alert being observed as true, what is the probability it was triggered by the peak time.



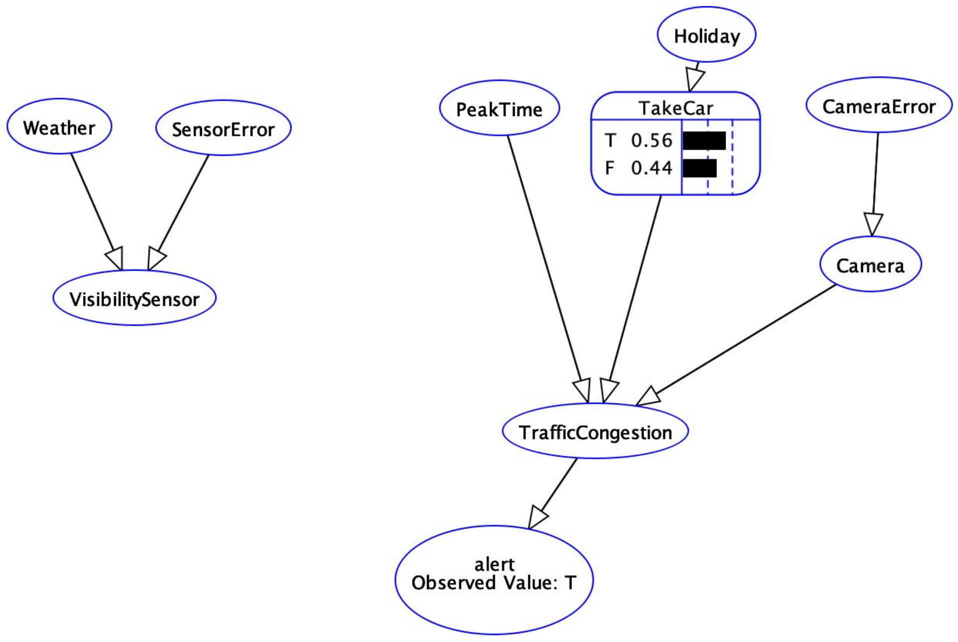
<bn1.xml>



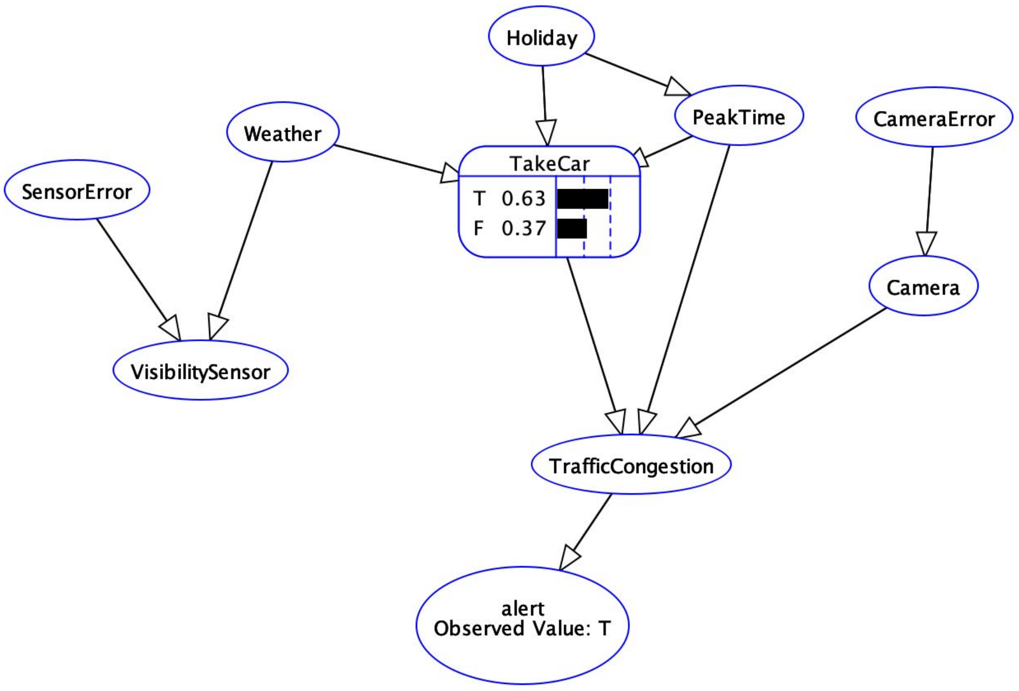
<bn2.xml>

Here we observe the effect of adding new connections between PeakTime and Holiday, and PeakTime and TakeCar.

This time, let’s do the diagnosis with the other feature. With the alert being observed as true, what is the probability it was triggered by the feature TakeCar.



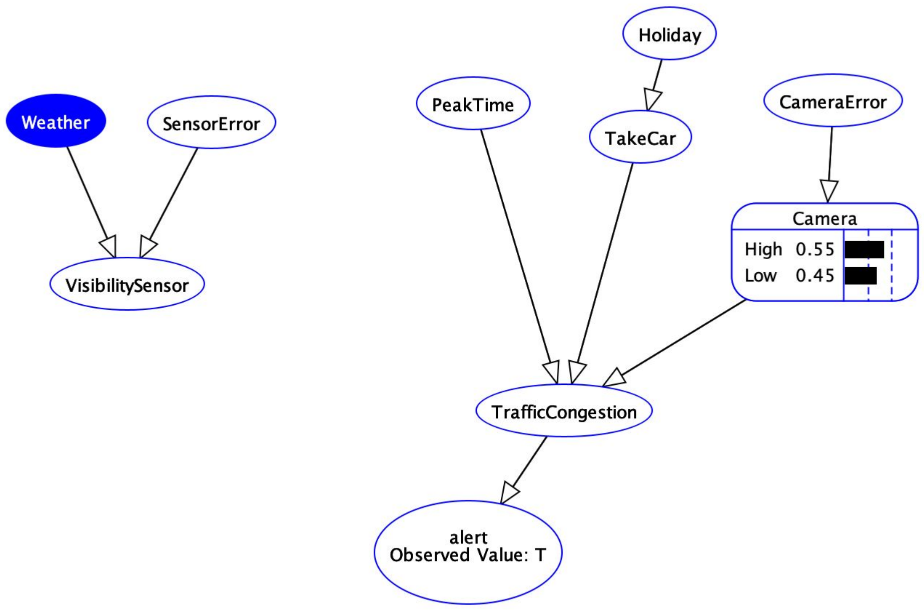
<bn1.xml>



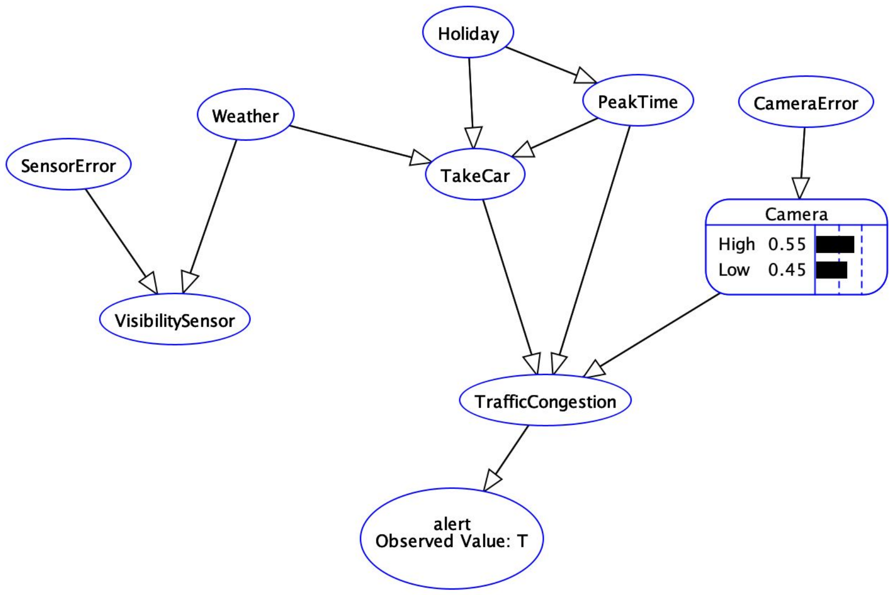
<bn2.xml>

Again, we observe the effect of adding new connections to the TakeCar feature.

Now, let’s do the diagnosis once more with the Camera feature, which is the feature that did not changed.



<bn1.xml>

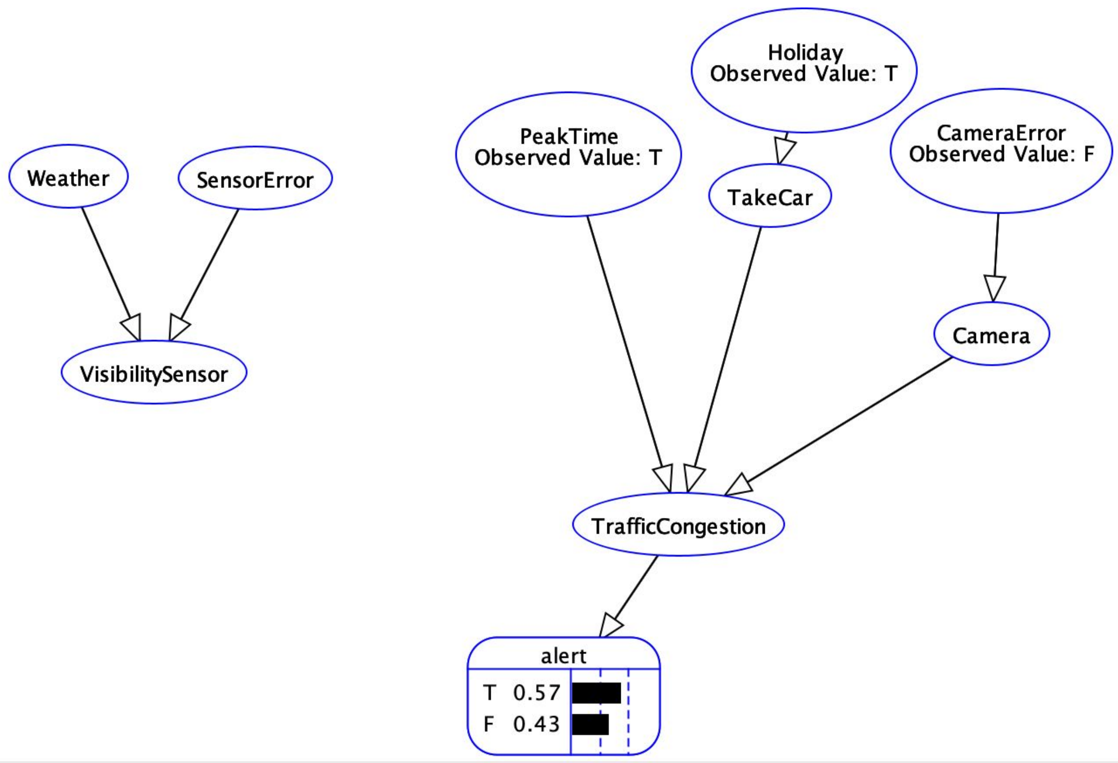


<bn2.xml>

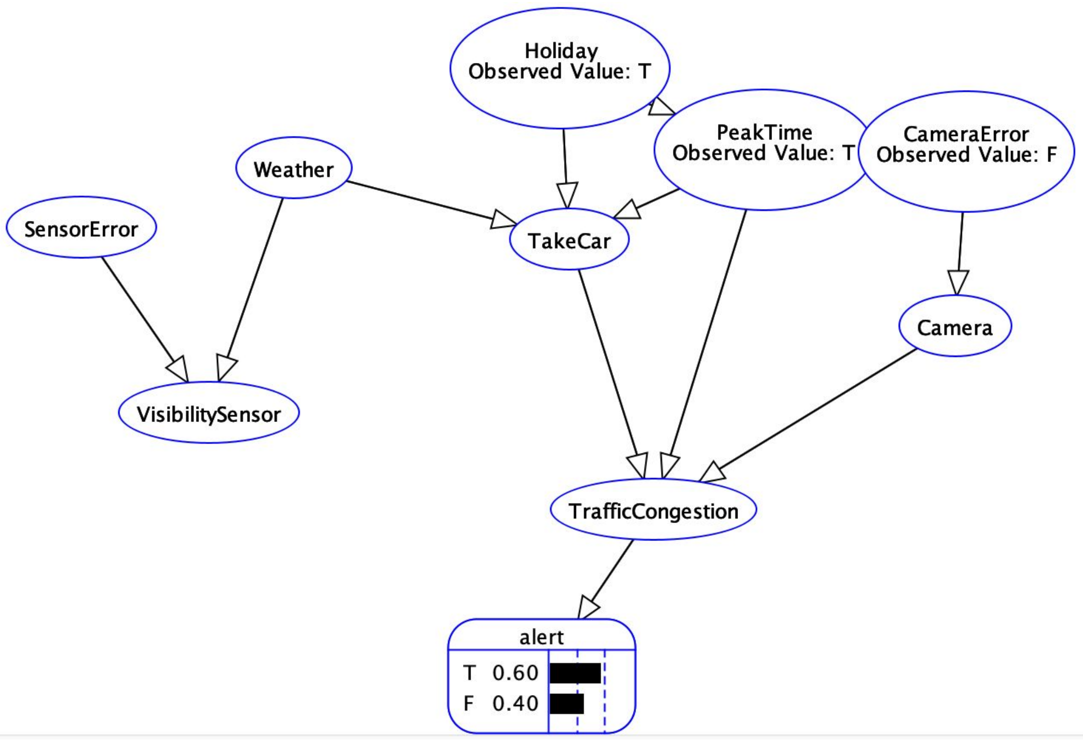
As you could see above, the probabilities did not changed. And it is possible to say that it is because that I did not add new connections to the Camera node.

B. Predictive

Given that it is peak time now, it is a holiday, and the camera does not give a wrong value, what are the chances an alert will be raised?



<bn1.xml>

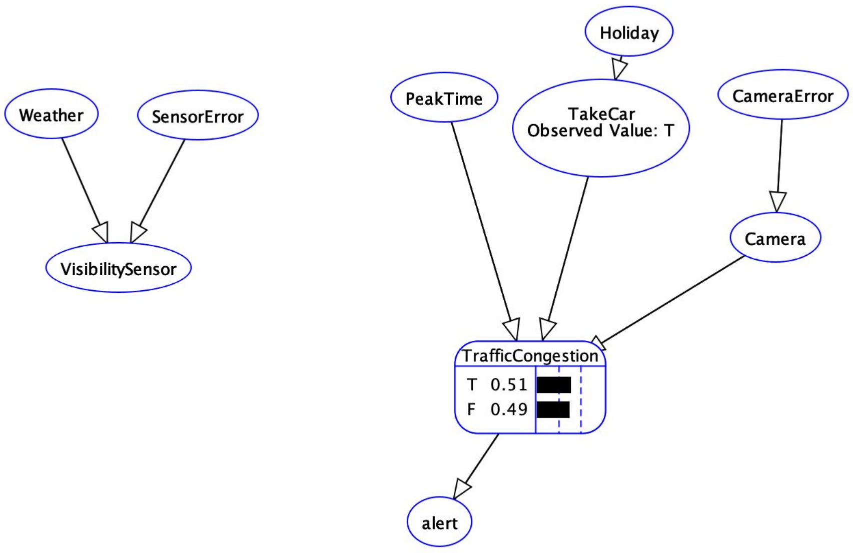


<bn2.xml>

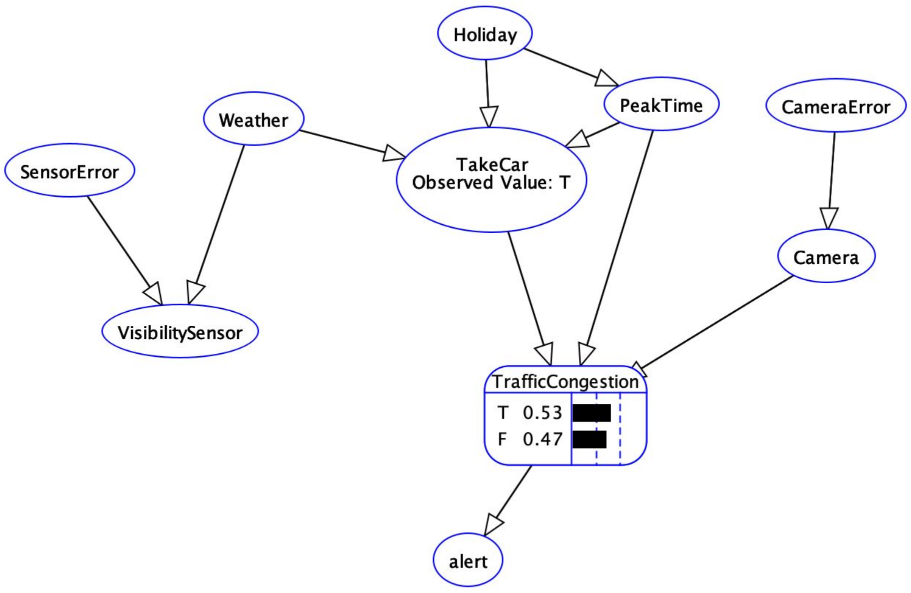
Here it can be observed that in BN1, there is a 57% chance of an alert will be raised given the observed features. However, in BN2, there is a 60% chance of an alert being raised. This is because of the new connections that I added in the BN2.

C. Profiling

With no observation, the probability that the traffic congestion is happening in BN1 is 44% and 46% in BN2. By observing the TakeCar feature as true, the probability of the traffic congestion rockets up to 51% in BN1 and 53% in BN2.

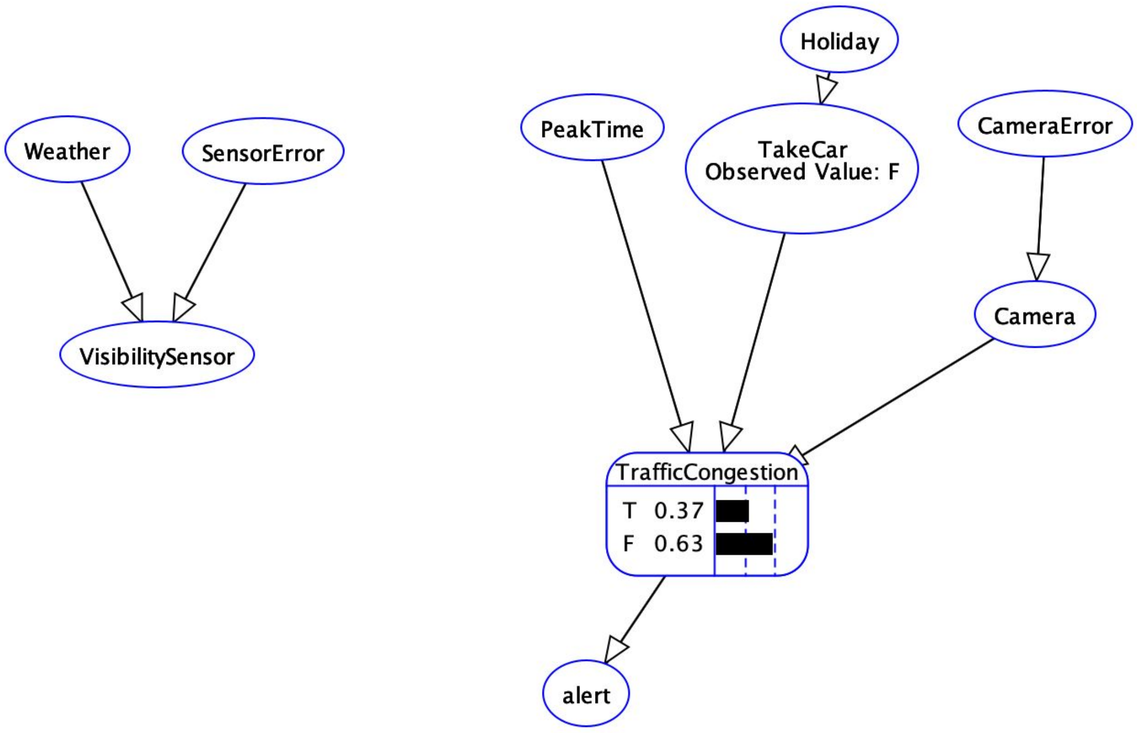


<bn1.xml>

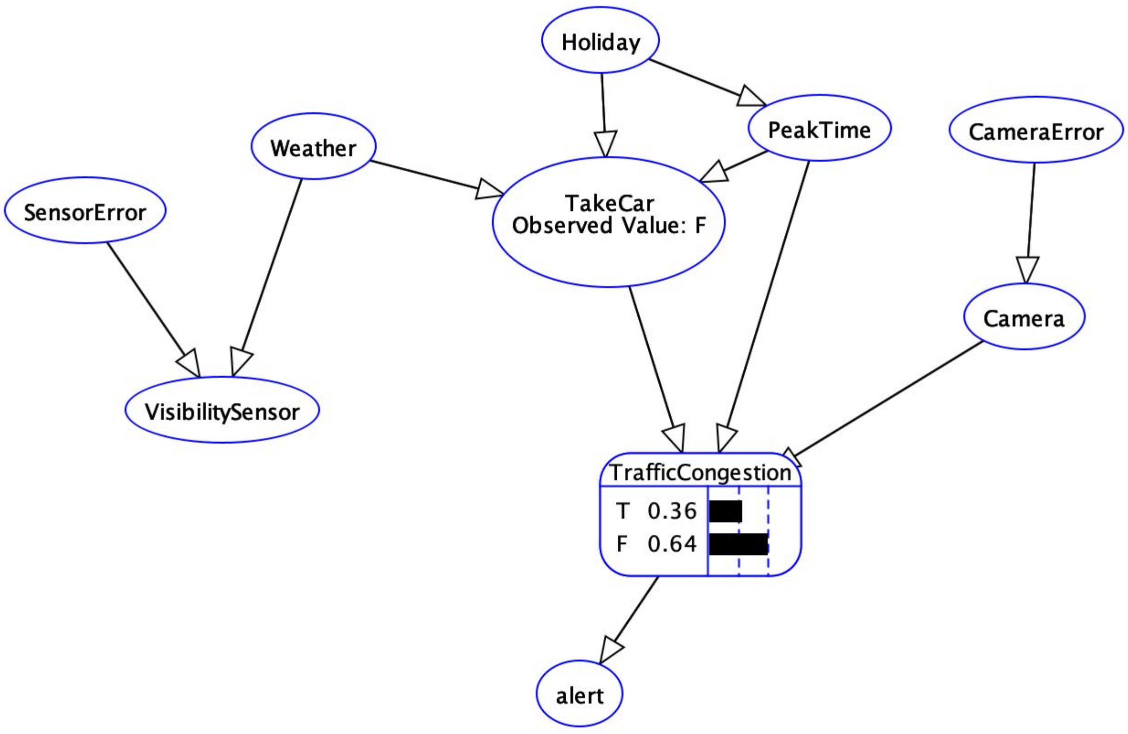


<bn2.xml>

However, by observing the TakeCar feature as false, the probability of traffic congestion decreases from 44% to 37% in BN1, and decreases from 46% to 36% in BN2.



<bn1.xml>



<bn2.xml>

Thus, it is possible to say that the TakeCar feature is the one who affects the most to the probability of the traffic congestion.

2- 2. Part 2

The aim of the part 2 is implement the variable elimination algorithm for inference by using Java. According to the specification, we could either use XMLParser to read and parse the xml file or hardcode the bayesian networks. I thought that it would be better to use the XMLParser, because by using the XMLParser I could easily test the program with many xml files that contain different bayesian networks.

After parsing the xml file, the program will build the graph by checking the connections. If you see the BayesianNetworkNode class, you could see the attribute “children”, “given” and “values”. The attribute “children” is an ArrayList of BayesianNetworkNode instances, and it stores all child nodes. The attribute “values” is an ArrayList of double type variables, where each double type variable stores the probability value. And the attribute “given” is an ArrayList of string, where each string is a name of the parent node. By using the “given” and “children”, the program could find the parent node and child node of the current node. And by using the attribute “values”, the program could get the list of probabilities of the current node.

In the BayesianNetworkParser class, you could see the method called “makeInference”. Basically, this method calls itself recursively to eliminate the variables one by one by parsing top nodes to the bottom nodes. It will start from the root node, and it will eliminate the variable one by one from root node to leaf node.

The makeInference() method uses the method called “generatePermutations”, which generates the permutations of the probabilities. For example, let’s assume that we have 3 variables a, b and c, where a and b are parent nodes of c. The feature a has 2 probabilities P(a=T) and P(a=F), and the feature b has 2 probabilities P(b=T) and P(b=F). By using the generatePermutations(), the program will get the list that contains all probabilities of the new table over the union of the a, b and c (i.e. P(a=T) * P(b=T) * P(c=T | a=T, b=T)). By using the permutation list, the makeInference() method could perform the variable elimination to make inference.

2 - 3. Part 3 (Extension)

For the extension, I made my Java program to handle all singly connected networks with the predictive query. Moreover, it could parse the multiply connected networks if the network if the network does not have a cycle in it. Since I am using the recursive methods to make inference and generate permutations list, my program for part 2 was able to handle all acyclic bayesian networks.

To be honest, I wanted to make my program to handle cyclic networks as well as acyclic networks, however, I was not able to find the suitable inference algorithm for the cyclic connected network due to the time constraints. Basically, I was running out of time, since I have 3 deadlines on week 10.

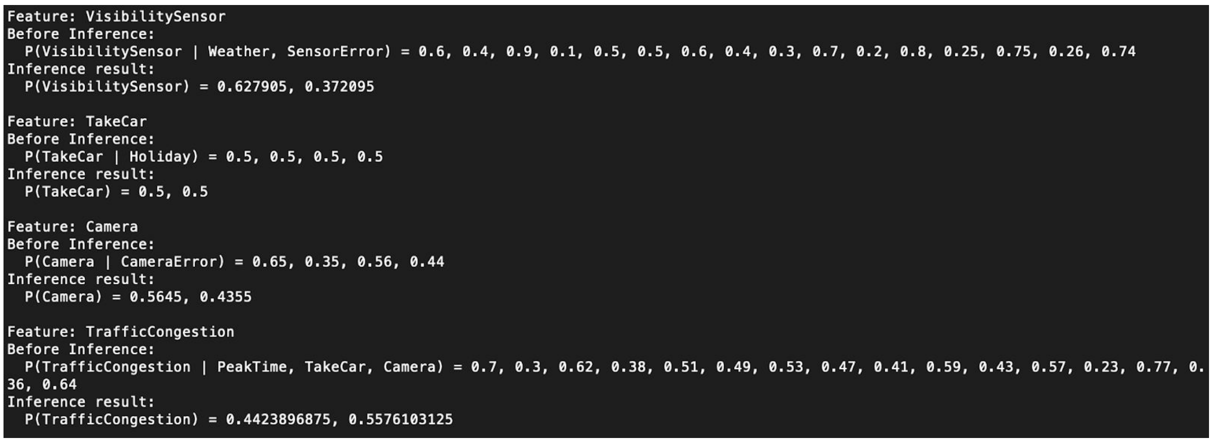
3. Evaluation

For the design of the bayesian networks in part 1, I am pretty confident with them. I represented all factors in the given scenario. Also, I represented all dependencies between factors by connecting nodes.

For the part 2 and part 3, my Java program actually works perfectly fine. It rarely has the Java floating point issue (expected 0.01 but it returns 0.0100000000002), however, it is just a small rounding difference. The program could perform the predictive query with all acyclic networks within reasonable time. Thus, I could say that my program is good enough for this practical.

4. Testing

Below is the result of “java A3main”.

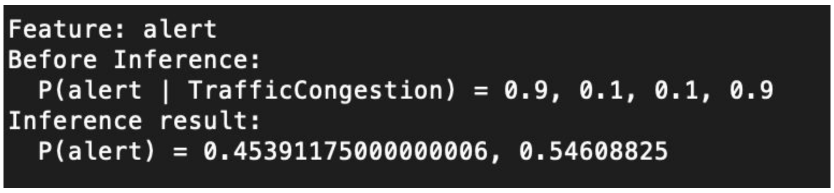


```
Feature: VisibilitySensor
Before Inference:
  P(VisibilitySensor | Weather, SensorError) = 0.6, 0.4, 0.9, 0.1, 0.5, 0.5, 0.6, 0.4, 0.3, 0.7, 0.2, 0.8, 0.25, 0.75, 0.26, 0.74
Inference result:
  P(VisibilitySensor) = 0.627905, 0.372095

Feature: TakeCar
Before Inference:
  P(TakeCar | Holiday) = 0.5, 0.5, 0.5, 0.5
Inference result:
  P(TakeCar) = 0.5, 0.5

Feature: Camera
Before Inference:
  P(Camera | CameraError) = 0.65, 0.35, 0.56, 0.44
Inference result:
  P(Camera) = 0.5645, 0.4355

Feature: TrafficCongestion
Before Inference:
  P(TrafficCongestion | PeakTime, TakeCar, Camera) = 0.7, 0.3, 0.62, 0.38, 0.51, 0.49, 0.53, 0.47, 0.41, 0.59, 0.43, 0.57, 0.23, 0.77, 0.
36, 0.64
Inference result:
  P(TrafficCongestion) = 0.4423896875, 0.5576103125
```



```
Feature: alert
Before Inference:
  P(alert | TrafficCongestion) = 0.9, 0.1, 0.1, 0.9
Inference result:
  P(alert) = 0.45391175000000006, 0.54608825
```

As you could see, the program could perform the predictive inference properly by using the variable elimination. Furthermore, you could run this program with bn1.xml or bn2.xml by using “java A3main <file_path>”.

5. Conclusion

This practical has been a great way to gain a practical understanding of how Bayesian Networks function, and how they can be a useful tool in working with conditional probabilities. I am confident in the design of my networks, as it shows a clear logical connection between all elements. Furthermore, the program that I wrote for part 2 works perfectly fine with variable eliminations for predictive inference. It could parse and handle all singly connected networks. By implementing the variable elimination program with Java, I was able to understand deeply about inference algorithms, which was definitely helpful. As I mentioned before, I was having hard time due to the time constraints. If I had more time to work on this practical, I would definitely try to improve my program to handle cyclic networks for predictive query.