INTRODUCTION

This report covers solving the Bayesian network for the canal problem in part one and implements a merging algorithm for two Bayesian networks.

The folder contains two created networks CANAL1 and CANAL2, in the XML format for part 1 problem. And in the report are shown four different queries that were done on each network.

The algorithm for networks merging is briefly explained in the report, and the results for merging three different kinds of networks are presented. The working of the algorithm is further described in the code's comments.

Running the code instructions

To run the merging algorithm, navigate to the src folder in the command window and write *java main BN1 BN2* where BN1 and BN2 represent the names of the XML files (type without .xml) in the networks folder.

Available files in the networks folder: example1, example2, CANAL1merge, CANAL2merge, alarm1, alarm2

1 DESIGN, IMPLEMENTATION AND EVALUATION

1.1 Part 1

AIspace program was used to create the Bayesian network to predict the risk of troubles on the Mecynian canal. The final design of the two networks can be found in the included folder under the names CANAL1 and CANAL2.

First, the CANAL1 model was created. Subsequently, there are named and further discussed individual nodes of the network.

Giant ship on the canal node. Since this is the biggest and most important canal, it is expected that a lot of traffic goes through and therefore, it was deduced that there is a 90% probability that the Giant Ship is passing through the canal.

Bad weather node. According to the map, the location of the Republic of Mecyna is near the equator. Therefore, it is assumed that the weather is influenced by the dry and rainy season. It is expected that during the rainy season, there often can come heavy rainfalls that influence the visibility. Since the rainy season usually last half of the year and also other weather factors are considered, it seems reasonable to deduce that there is a 35% chance of having bad weather.

The weather status is determined by the **weather sensor** that can give a wrong prediction with a chance of 1%, and thus in 99% of cases, the sensor conveys the true information.

For other threat nodes, **Tide** and **Canal Busy,** the probabilities were implicitly given in the specifications. Low tide – 6 hours a day equals to 6/24 probability of the low tide. The canal is busy 1/5 of the year gives a 20% probability that the canal is busy.

Traffic sensor node – is used basically to check whether to reduce or not the speed of the traffic. There are no big weights on this factor; thus, it has not a big impact on the decision. Its parent node is **Canal Busy,** and the sensor reads the traffic with 90% accuracy.

Conditional probability in the **Risk Giant Ship Get Stuck node** – when there is no giant ship, there is no chance that it gets stuck; thus, the probability will always be lower than 3%. The probability of the ship getting stuck goes above 3% when there is a giant ship on the canal, and two or more of the threats are true.

Alert node is true when the risk is more than 3%.

Reduce speed node consists of conditional probability – when the alert is true, the speed is almost certainly reduced, the traffic sensor does not have a big influence, as mentioned before.

For the second task, the CANAL1 model was augmented by two new events. The first added event is **construction work** that can occur throughout the year with 10% probability, and it contributes to the **risk node** conditional probability. Those could be any modifications of the canal or repairs. The probability for the construction to be true was estimated at 10%.

The second added event is a **police warning** that directly influences Speed reduction decision. When it is true, the speed is almost certainly reduced even if the alarm is false.

Queries – Evaluation

Both networks were tested by making various observations and then compared to the idea that the networks are based on. For example, as shown in figure 1 where the variable of the Giant Ship is set to True as well as two threats such as Weather to Bad and Tide to Low. The expected output was then compared to the output received from the network. In this case, the expected output is that they will most likely reduce the speed on the canal since the two threats are already true. This was confirmed by the output from the network that can be seen in the figure.

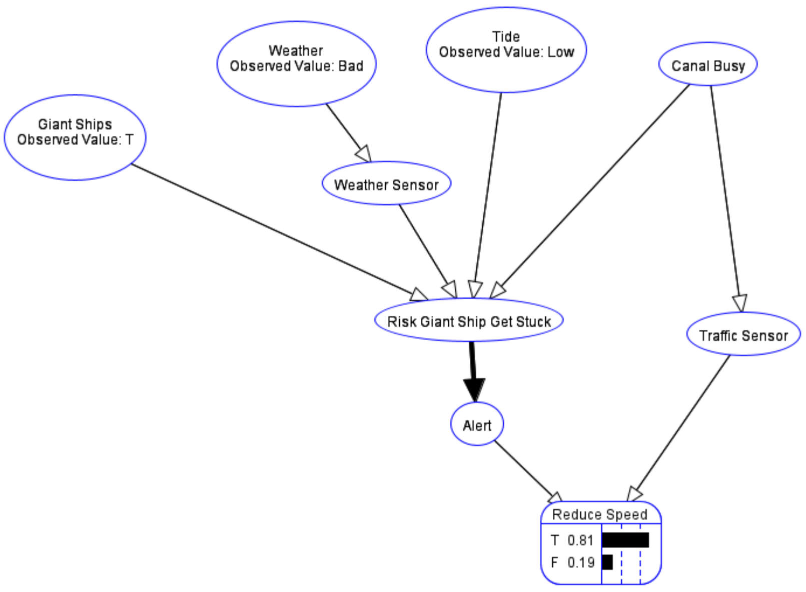


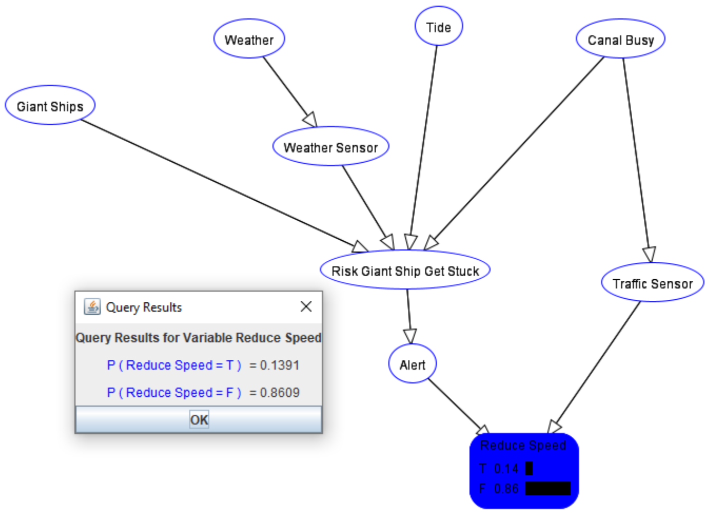
Figure 1: Evaluation according to the expected output

Besides, the four different types of queries were made on both networks, and the differences were further discussed.

1) Predictive query

The first type of query is predictive. It is desirable to find out the probability that the government will decide to reduce the speed on the canal, which is the direction of reasoning down through the network.

At first, the probabilities without any observations were investigated in both networks. The result from the first network is presented in the figure below. The second's probability of reducing the speed being true was 20.018%. That is higher than in the first one, caused by additional threat (Construction Work) in the second network. The risk that the giant ship gets stuck is higher than three per cent when two or more threats are true; since there is an additional threat in the second network, it makes sense that the probability of reducing the speed gets higher. Also, the second network involves the option that the speed in the canal could be influenced by police warning, which could happen independently on the Alert.



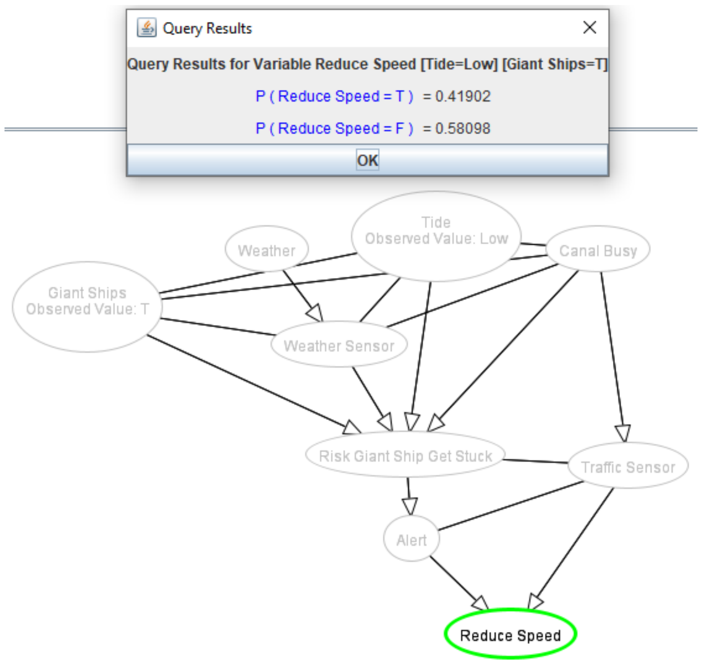
Query Results for Variable Reduce Speed

P (Reduce Speed = T) = 0.1391

P (Reduce Speed = F) = 0.8609

Figure 2: CANAL1 network without observations

The posterior probability of p(Reduce Speed | Giant Ships=True, Tide=Low) was investigated in both networks. There are two observations from which one is a threat (low tide) that increases the probability of a giant ship being stuck. With these observations, there is a much higher probability that the government will reduce the speed of the canal in both networks. The probability in the CANAL2 network is again higher for reasons mentioned before. The results are displayed in the figures below.

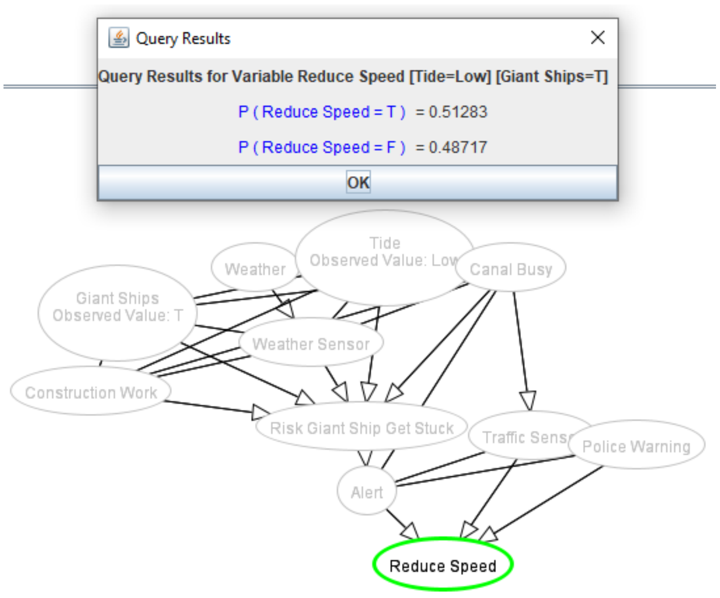


Query Results for Variable Reduce Speed [Tide=Low] [Giant Ships=T]

P (Reduce Speed = T) = 0.41902

P (Reduce Speed = F) = 0.58098

Figure 3: CANAL1 results


Query Results

Query Results for Variable Reduce Speed [Tide=Low] [Giant Ships=T]

P (Reduce Speed = T) = 0.51283

P (Reduce Speed = F) = 0.48717

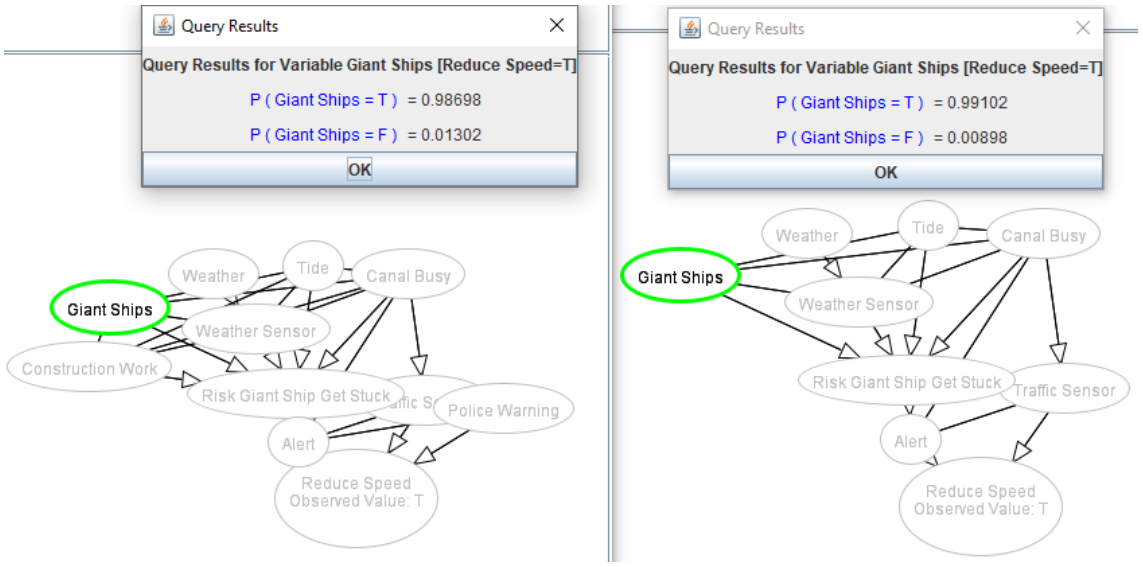
Figure 4: CANAL2 results

2) Diagnostic query

In this case, it is known that the speed on the canal was reduced, and it is desirable to find out the posterior probability of p(Giant Ship | Reduce Speed=True), which means the probability that the giant ship is on the canal given that we know that the speed of the canal was reduced. The direction of the reasoning is up from the bottom of the network.

In both cases, the probabilities without any observations are prior probabilities of a giant ship being on the canal that is true with 90% probability.

Now considering the observation that the speed was reduced, the probability of the giant ship being on the canal is slightly higher in the CANAL1 network because, in the CANAL2 network, the decision about reducing the speed is also dependent on the Police Warning, which can happen no matter whether the giant ship is on the canal or not.



Query Results

Query Results for Variable Giant Ships [Reduce Speed=T]

P (Giant Ships = T) = 0.98698

P (Giant Ships = F) = 0.01302

Query Results

Query Results for Variable Giant Ships [Reduce Speed=T]

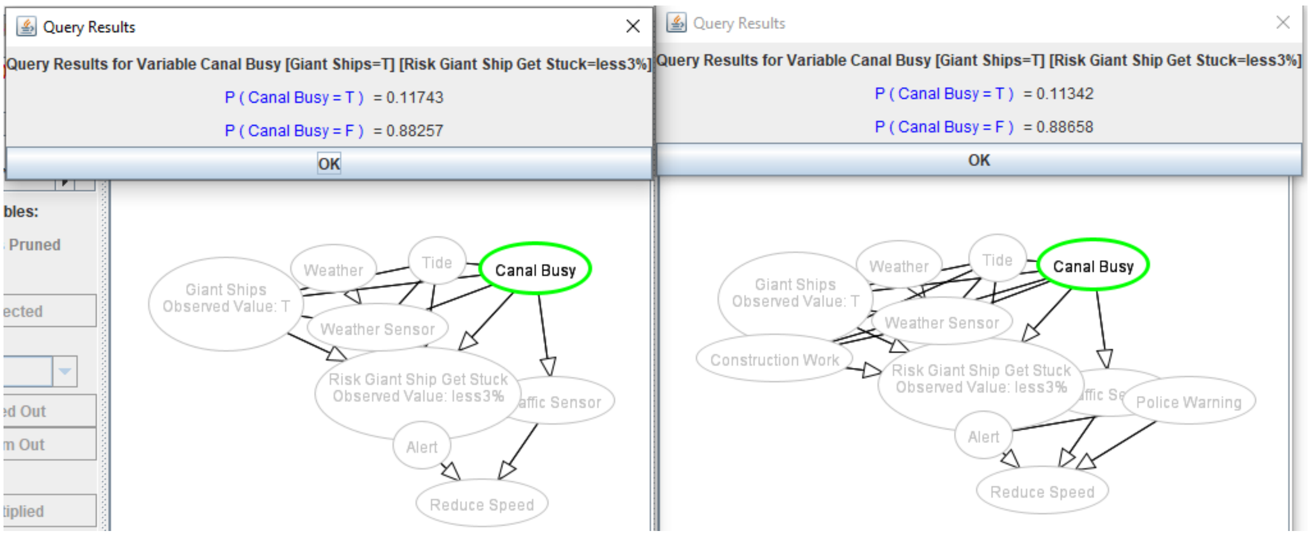
P (Giant Ships = T) = 0.99102

P (Giant Ships = F) = 0.00898

Figure 5: Diagnostic results – CANAL2 on the left, CANAL1 on the right

3) Intercausal query

In this query, the posterior probability of p(Canal Busy | Giant Ships Observed=True, Risk Giant Ship Get Stuck=less3%) which means "what is the probability of the canal being busy given that we observed that there is a giant ship on the canal and chance of getting stuck is less than 3%". The results are presented below. Again, because of the additional factor, the CANAL2 has a slightly lower probability of the canal being busy since another factor could cause risk going above 3%.



Query Results

Query Results for Variable Canal Busy [Giant Ships=T] [Risk Giant Ship Get Stuck=less3%]

P (Canal Busy = T) = 0.11743

P (Canal Busy = F) = 0.88257

Query Results

Query Results for Variable Canal Busy [Giant Ships=T] [Risk Giant Ship Get Stuck=less3%]

P (Canal Busy = T) = 0.11342

P (Canal Busy = F) = 0.88658

Figure 6: Intercausal results-CANAL1 and CANAL2

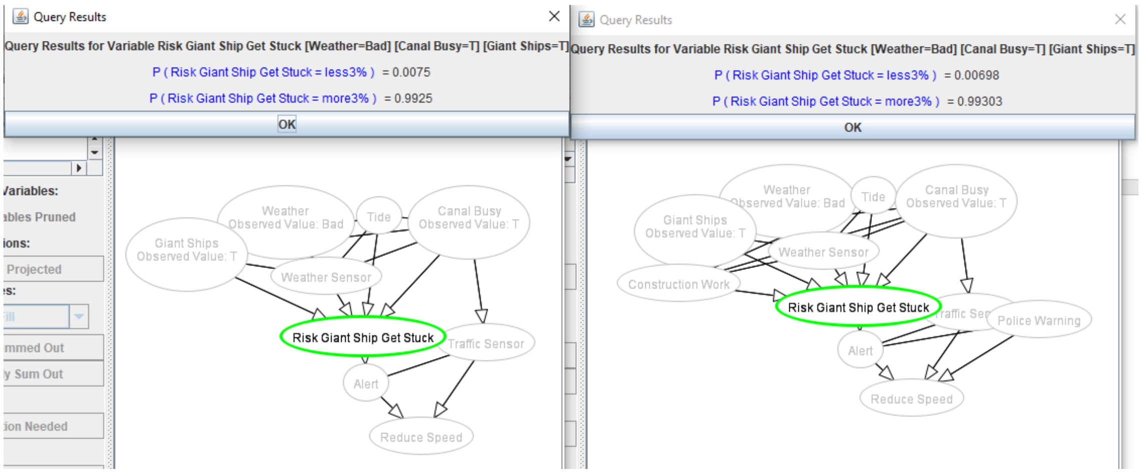
4) Profiling query

This query is much more investigative than others. This example investigates the probability that the risk of the giant ship getting stuck is above 3% while assigning different values (True/False) to the Canal Busy factor.

Without any observations, the probability of the risk being more than 3% is 13.88% in the CANAL1 and 18.38% in the CANAL2.

Now there were also added two other observations – Giant Ship=True and Weather=Bad. Different values were then assigned to the Canal Busy as mentioned, and it was observed how it changed the risk.

Firstly, Canal Busy value was set to True. It is basically certain that the risk is above 3%, which complies with the rule when there are two threats, then the risk is above 3%. The only ambiguity is caused by a weather sensor that could read a wrong value. CANAL2 has a slightly higher probability because it has one additional factor.


Query Results

Query Results for Variable Risk Giant Ship Get Stuck [Weather=Bad] [Canal Busy=T] [Giant Ships=T]
P (Risk Giant Ship Get Stuck = less3%) = 0.0075
P (Risk Giant Ship Get Stuck = more3%) = 0.9925

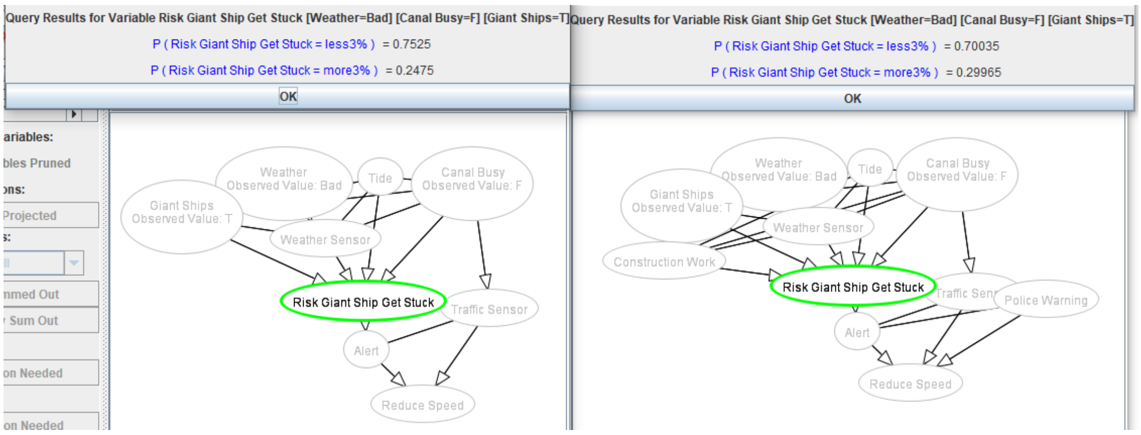
Query Results

Query Results for Variable Risk Giant Ship Get Stuck [Weather=Bad] [Canal Busy=T] [Giant Ships=T]
P (Risk Giant Ship Get Stuck = less3%) = 0.00698
P (Risk Giant Ship Get Stuck = more3%) = 0.99303

OK
Weather Observed Value: Bad
Tide
Canal Busy Observed Value: T
Giant Ships Observed Value: T
Weather Sensor
Construction Work
Risk Giant Ship Get Stuck
Traffic Sensor
Police Warning
Alert
Reduce Speed

Figure 7: Profiling query-when Canal Busy=True (CANAL1 and CANAL2 as follows)

The same was done for the Canal Busy value=False, and the results are presented below.


Query Results for Variable Risk Giant Ship Get Stuck [Weather=Bad] [Canal Busy=F] [Giant Ships=T]
P (Risk Giant Ship Get Stuck = less3%) = 0.7525
P (Risk Giant Ship Get Stuck = more3%) = 0.2475

Query Results for Variable Risk Giant Ship Get Stuck [Weather=Bad] [Canal Busy=F] [Giant Ships=T]
P (Risk Giant Ship Get Stuck = less3%) = 0.70035
P (Risk Giant Ship Get Stuck = more3%) = 0.29965

OK
Weather Observed Value: Bad
Tide
Canal Busy Observed Value: F
Giant Ships Observed Value: T
Weather Sensor
Construction Work
Risk Giant Ship Get Stuck
Traffic Sensor
Police Warning
Alert
Reduce Speed

Figure 8: Profiling query-when Canal Busy=False (CANAL1 and CANAL2 as follows)

1.2 Part 2

For part two, the merging algorithm was implemented. First, it accepts the names of two XML files containing two networks to be merged. These networks are then individually parsed into two HashMaps where the node's name is the key, and the value is NetworkNode object that stores the node's name, values, parents, and probability table.

The merging algorithm then categorizes the nodes as non-intersection, external and internal nodes. Non-intersection nodes are transferred to the merged network with properties from their previous network. Delete rule is applied on internal nodes. External nodes are merged, so the parents from both previous networks are preserved, and new conditional probabilities are calculated for them. These probabilities are calculated according to the number of parents from the previous networks. The maximum number of parents for the new network is five, and there are six possible ways of combining those parents. The algorithm picks the appropriate combination and calculates the probabilities accordingly, and normalizes those probabilities. There are manually inserted patterns of indexes that are used for conditional probability calculation. These patterns were derived on the paper.

After that new HashMap of the merged network is created, and its result is printed in the console.

Testing

For testing, some of the probability calculations done by the algorithm were compared to ones done manually. Also expected number and names of nodes were compared to the ones produced by the algorithm. The conditional probabilities computed for external nodes were checked whether they sum up to 1. The algorithm was mainly tested on the networks that are included in the folder named *networks*.

Evaluation

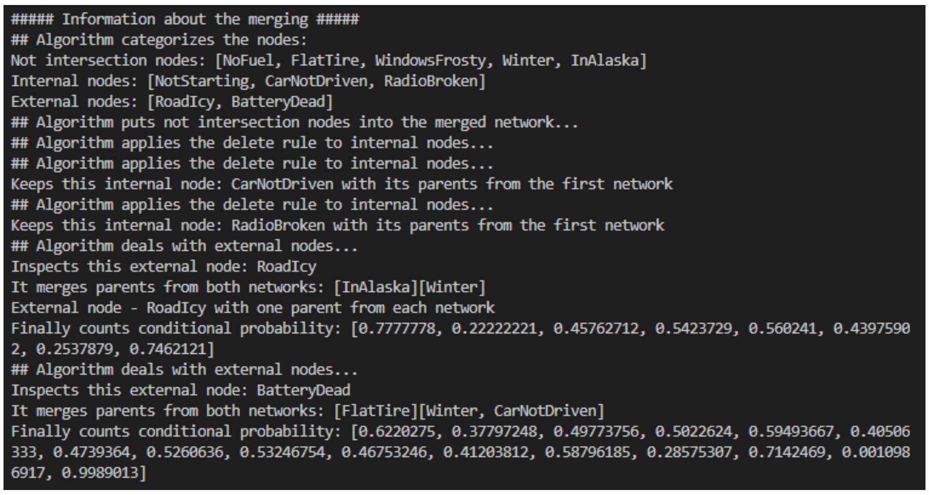
The algorithm is able to process XML files and store the nodes in the HasMap, which makes it easy to access the required node according to its name. Categorization of the nodes is straightforward, as well as dealing with non-intersection and internal nodes. It is more complicated when it gets to external nodes and combining probabilities.

It is not able to deal with the conditional probability of external nodes where both networks have the same internal node as a parent. It complies with simplifying assumption, and thus it cannot create more than five parents for one node. Manually typed patterns of indexes were used to calculate conditional probabilities. For future development, there could be a function that would calculate these patterns based on a number of parents and calculate the possible combinations of their values.

Examples

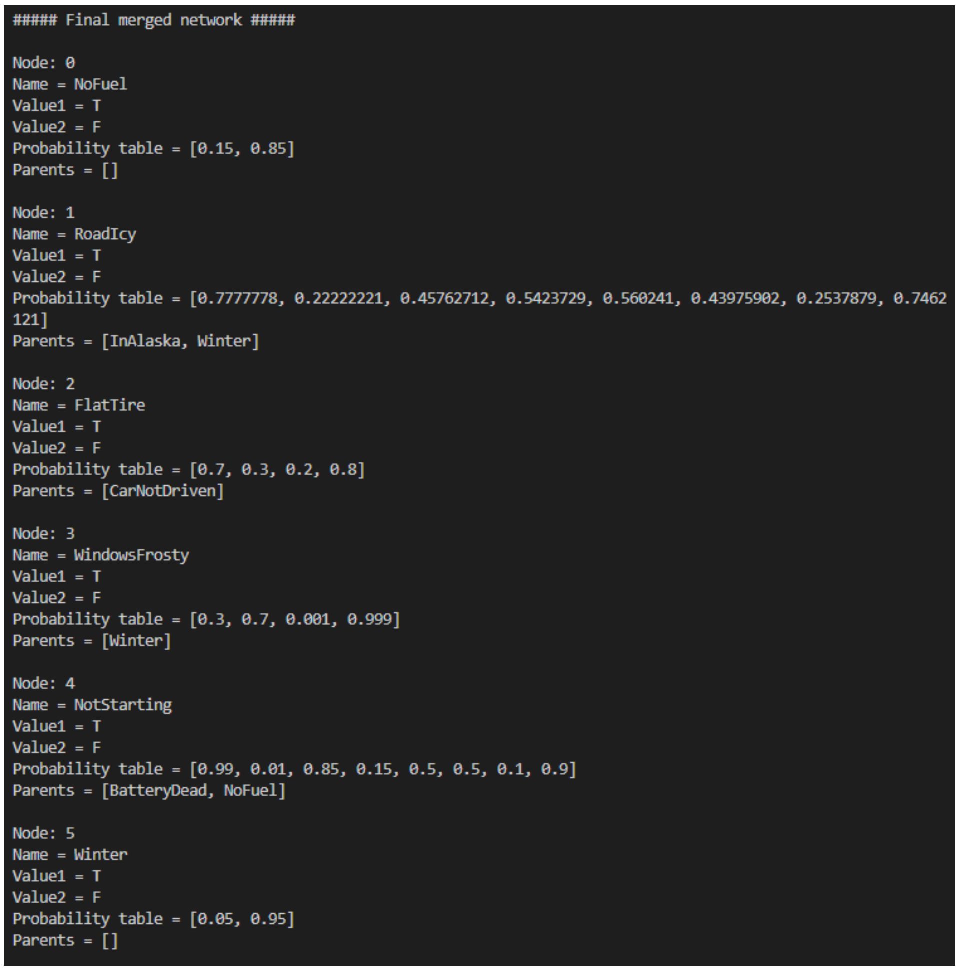
Here are the presented results of merging three different kinds of networks mentioned in the testing above. The screenshots contain the results printed in the console after merging.

First two networks example1 and example2.



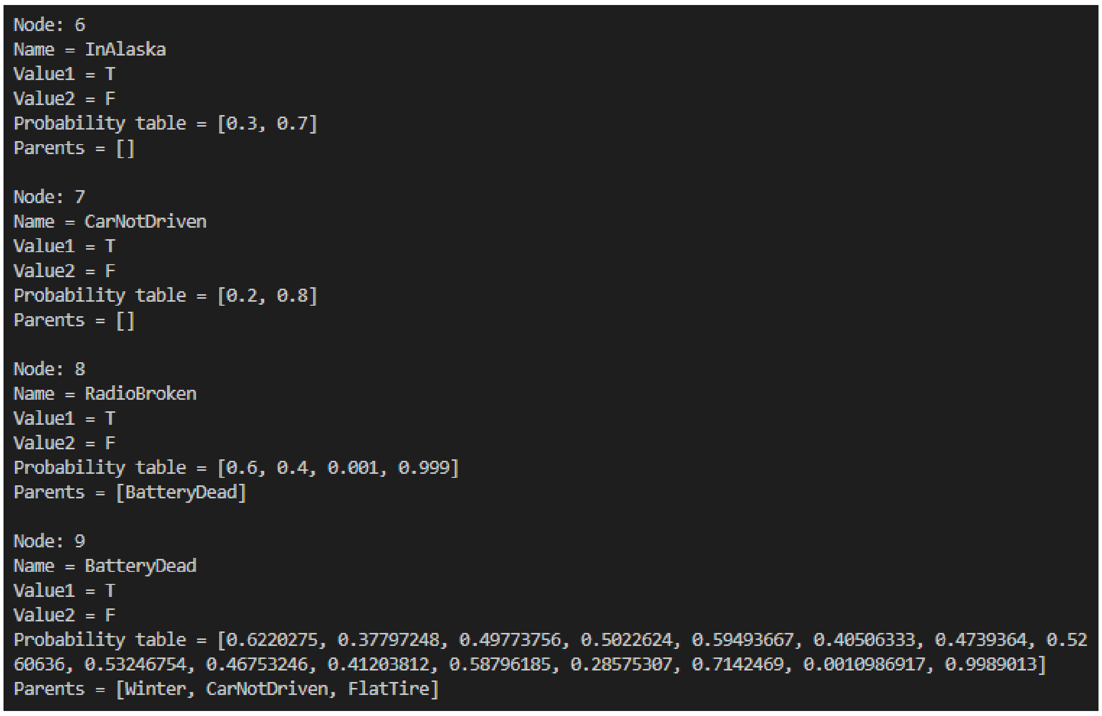
```
##### Information about the merging #####
## Algorithm categorizes the nodes:
Not intersection nodes: [NoFuel, FlatTire, WindowsFrosty, Winter, InAlaska]
Internal nodes: [NotStarting, CarNotDriven, RadioBroken]
External nodes: [RoadIcy, BatteryDead]
## Algorithm puts not intersection nodes into the merged network...
## Algorithm applies the delete rule to internal nodes...
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: CarNotDriven with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: RadioBroken with its parents from the first network
## Algorithm deals with external nodes...
Inspects this external node: RoadIcy
It merges parents from both networks: [InAlaska][Winter]
External node - RoadIcy with one parent from each network
Finally counts conditional probability: [0.7777778, 0.22222221, 0.45762712, 0.5423729, 0.560241, 0.4397590
2, 0.2537879, 0.7462121]
## Algorithm deals with external nodes...
Inspects this external node: BatteryDead
It merges parents from both networks: [FlatTire][Winter, CarNotDriven]
Finally counts conditional probability: [0.6220275, 0.37797248, 0.49773756, 0.5022624, 0.59493667, 0.40506
333, 0.4739364, 0.5260636, 0.53246754, 0.46753246, 0.41203812, 0.58796185, 0.28575307, 0.7142469, 0.001098
6917, 0.9989013]
```

Figure 9: Merging information – example1 and example2



```
##### Final merged network #####

Node: 0
Name = NoFuel
Value1 = T
Value2 = F
Probability table = [0.15, 0.85]
Parents = []

Node: 1
Name = RoadIcy
Value1 = T
Value2 = F
Probability table = [0.7777778, 0.22222221, 0.45762712, 0.5423729, 0.560241, 0.43975902, 0.2537879, 0.7462
121]
Parents = [InAlaska, Winter]

Node: 2
Name = FlatTire
Value1 = T
Value2 = F
Probability table = [0.7, 0.3, 0.2, 0.8]
Parents = [CarNotDriven]

Node: 3
Name = WindowsFrosty
Value1 = T
Value2 = F
Probability table = [0.3, 0.7, 0.001, 0.999]
Parents = [Winter]

Node: 4
Name = NotStarting
Value1 = T
Value2 = F
Probability table = [0.99, 0.01, 0.85, 0.15, 0.5, 0.5, 0.1, 0.9]
Parents = [BatteryDead, NoFuel]

Node: 5
Name = Winter
Value1 = T
Value2 = F
Probability table = [0.05, 0.95]
Parents = []
```

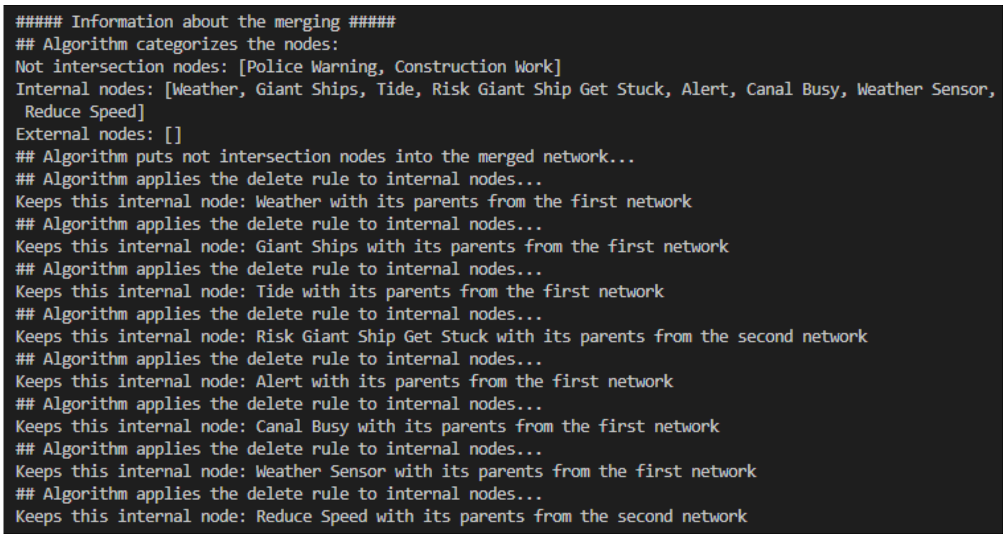
Figure 10: The first part of the merged network – example1 and example2



```
Node: 6
Name = InAlaska
Value1 = T
Value2 = F
Probability table = [0.3, 0.7]
Parents = []

Node: 7
Name = CarNotDriven
Value1 = T
Value2 = F
Probability table = [0.2, 0.8]
Parents = []

Node: 8
Name = RadioBroken
Value1 = T
Value2 = F
Probability table = [0.6, 0.4, 0.001, 0.999]
Parents = [BatteryDead]

Node: 9
Name = BatteryDead
Value1 = T
Value2 = F
Probability table = [0.6220275, 0.37797248, 0.49773756, 0.5022624, 0.59493667, 0.40506333, 0.4739364, 0.52
60636, 0.53246754, 0.46753246, 0.41203812, 0.58796185, 0.28575307, 0.7142469, 0.0010986917, 0.9989013]
Parents = [Winter, CarNotDriven, FlatTire]
```

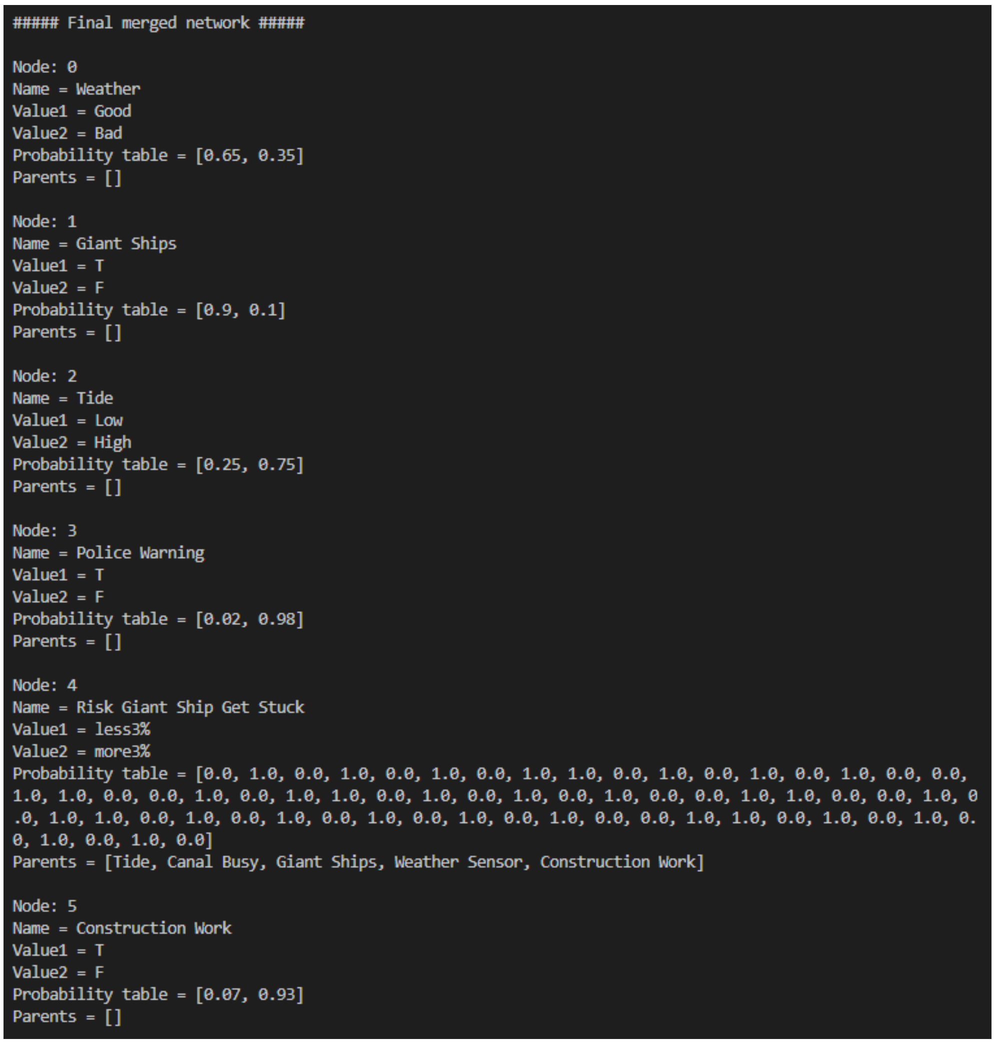
Figure 11: The second part of the merged network – example1 and example2

The second example is CANAL1merged and CANAL2merged files—networks from the first part, which were modified to comply with simplifications.



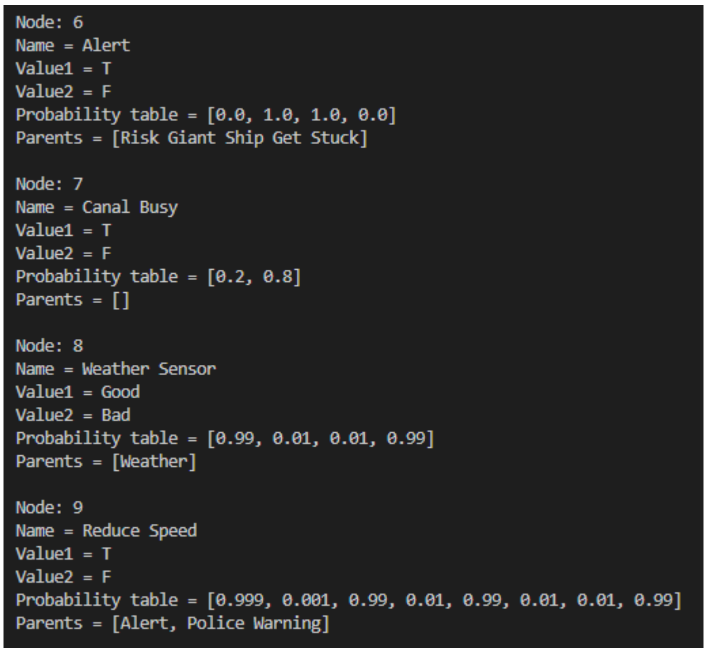
```
##### Information about the merging #####
## Algorithm categorizes the nodes:
Not intersection nodes: [Police Warning, Construction Work]
Internal nodes: [Weather, Giant Ships, Tide, Risk Giant Ship Get Stuck, Alert, Canal Busy, Weather Sensor,
 Reduce Speed]
External nodes: []
## Algorithm puts not intersection nodes into the merged network...
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Weather with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Giant Ships with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Tide with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Risk Giant Ship Get Stuck with its parents from the second network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Alert with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Canal Busy with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Weather Sensor with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: Reduce Speed with its parents from the second network
```

Figure 12: Merging information - CANAL1merge and CANAL2merge



```
##### Final merged network #####

Node: 0
Name = Weather
Value1 = Good
Value2 = Bad
Probability table = [0.65, 0.35]
Parents = []

Node: 1
Name = Giant Ships
Value1 = T
Value2 = F
Probability table = [0.9, 0.1]
Parents = []

Node: 2
Name = Tide
Value1 = Low
Value2 = High
Probability table = [0.25, 0.75]
Parents = []

Node: 3
Name = Police Warning
Value1 = T
Value2 = F
Probability table = [0.02, 0.98]
Parents = []

Node: 4
Name = Risk Giant Ship Get Stuck
Value1 = less3%
Value2 = more3%
Probability table = [0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0,
1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0
.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.
0, 1.0, 0.0, 1.0, 0.0]
Parents = [Tide, Canal Busy, Giant Ships, Weather Sensor, Construction Work]

Node: 5
Name = Construction Work
Value1 = T
Value2 = F
Probability table = [0.07, 0.93]
Parents = []
```

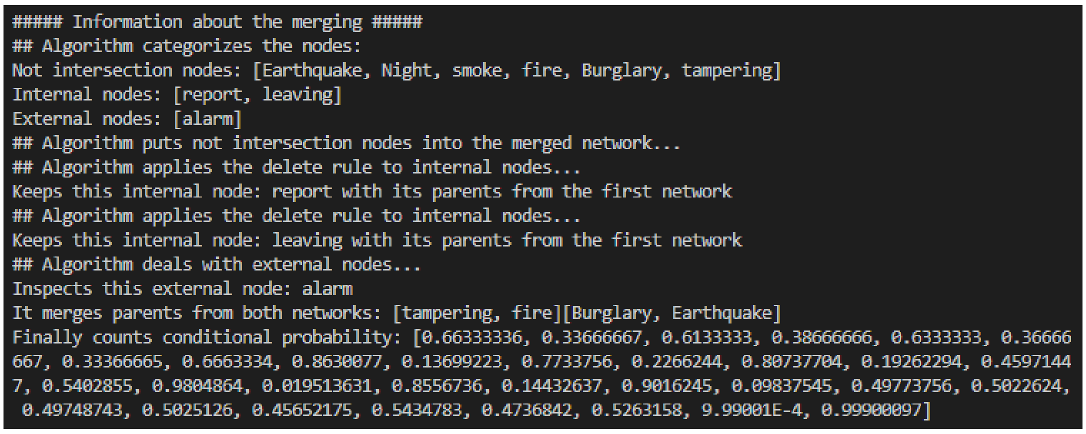
Figure 13: The first part of the merged network - CANAL1merge and CANAL2merge



```
Node: 6
Name = Alert
Value1 = T
Value2 = F
Probability table = [0.0, 1.0, 1.0, 0.0]
Parents = [Risk Giant Ship Get Stuck]

Node: 7
Name = Canal Busy
Value1 = T
Value2 = F
Probability table = [0.2, 0.8]
Parents = []

Node: 8
Name = Weather Sensor
Value1 = Good
Value2 = Bad
Probability table = [0.99, 0.01, 0.01, 0.99]
Parents = [Weather]

Node: 9
Name = Reduce Speed
Value1 = T
Value2 = F
Probability table = [0.999, 0.001, 0.99, 0.01, 0.99, 0.01, 0.01, 0.99]
Parents = [Alert, Police Warning]
```

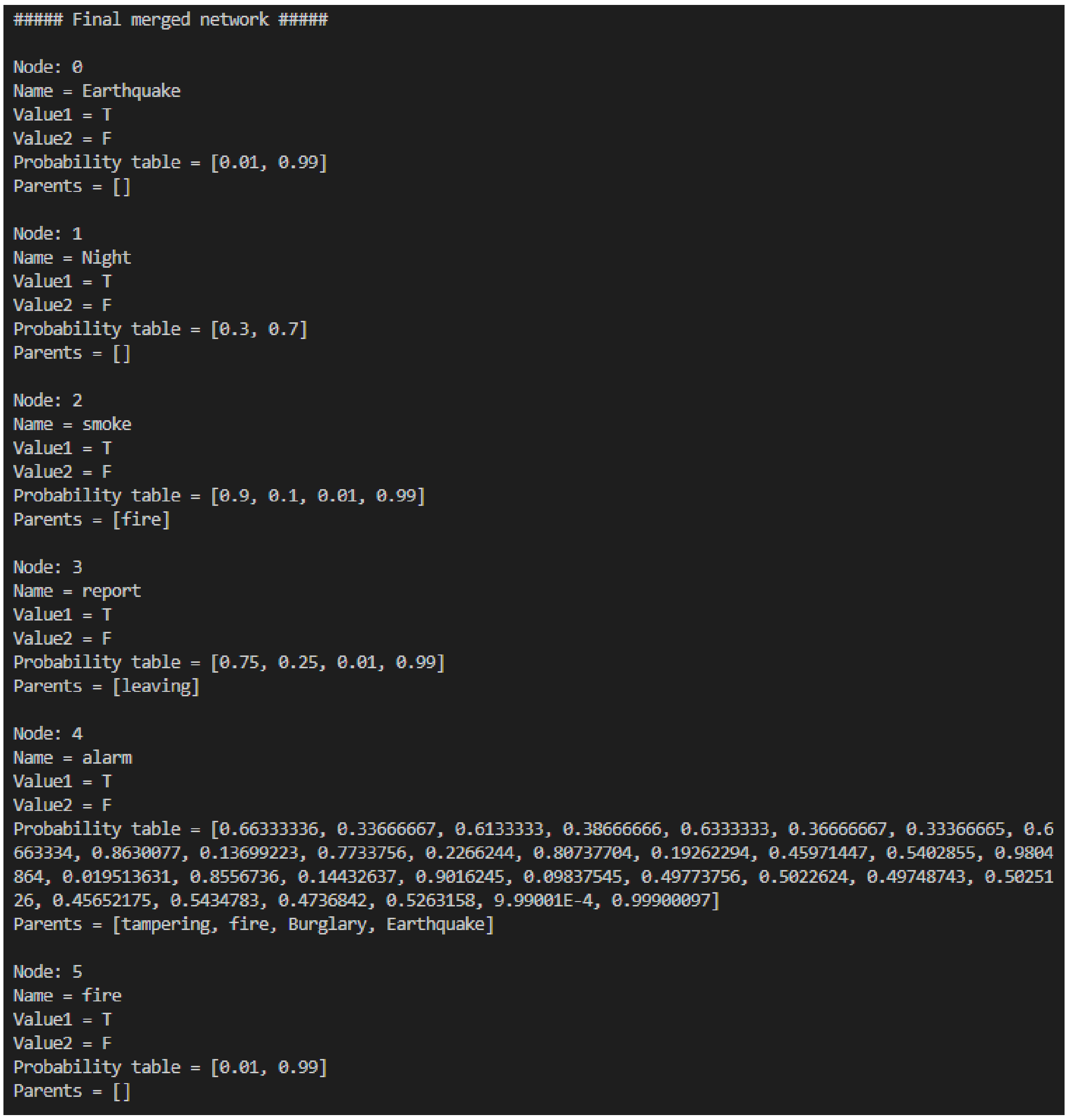
Figure 14: The second part of the merged network - CANAL1merge and CANAL2merge

As a third example, it was loaded a sample network from AIspace and saved as alarm1 and then modified and saved as alarm2.



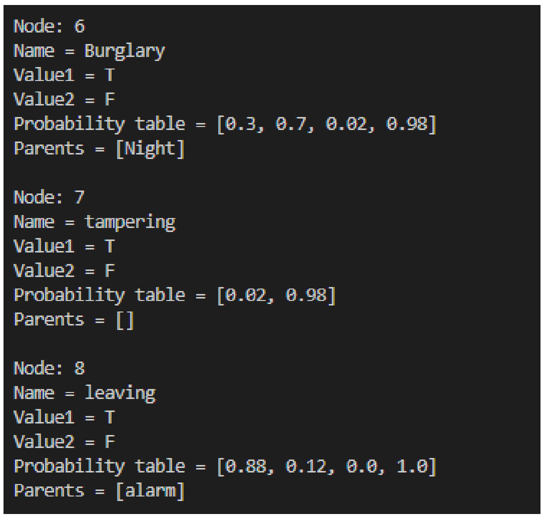
```
##### Information about the merging #####
## Algorithm categorizes the nodes:
Not intersection nodes: [Earthquake, Night, smoke, fire, Burglary, tampering]
Internal nodes: [report, leaving]
External nodes: [alarm]
## Algorithm puts not intersection nodes into the merged network...
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: report with its parents from the first network
## Algorithm applies the delete rule to internal nodes...
Keeps this internal node: leaving with its parents from the first network
## Algorithm deals with external nodes...
Inspects this external node: alarm
It merges parents from both networks: [tampering, fire][Burglary, Earthquake]
Finally counts conditional probability: [0.66333336, 0.33666667, 0.6133333, 0.38666666, 0.6333333, 0.36666
667, 0.33366665, 0.6663334, 0.8630077, 0.13699223, 0.7733756, 0.2266244, 0.80737704, 0.19262294, 0.4597144
7, 0.5402855, 0.9804864, 0.019513631, 0.8556736, 0.14432637, 0.9016245, 0.09837545, 0.49773756, 0.5022624,
 0.49748743, 0.5025126, 0.45652175, 0.5434783, 0.4736842, 0.5263158, 9.99001E-4, 0.99900097]
```

Figure 15: Merging information – alarm1 and alarm2



```
##### Final merged network #####

Node: 0
Name = Earthquake
Value1 = T
Value2 = F
Probability table = [0.01, 0.99]
Parents = []

Node: 1
Name = Night
Value1 = T
Value2 = F
Probability table = [0.3, 0.7]
Parents = []

Node: 2
Name = smoke
Value1 = T
Value2 = F
Probability table = [0.9, 0.1, 0.01, 0.99]
Parents = [fire]

Node: 3
Name = report
Value1 = T
Value2 = F
Probability table = [0.75, 0.25, 0.01, 0.99]
Parents = [leaving]

Node: 4
Name = alarm
Value1 = T
Value2 = F
Probability table = [0.66333336, 0.33666667, 0.6133333, 0.38666666, 0.6333333, 0.36666667, 0.33366665, 0.6
663334, 0.8630077, 0.13699223, 0.7733756, 0.2266244, 0.80737704, 0.19262294, 0.45971447, 0.5402855, 0.9804
864, 0.019513631, 0.8556736, 0.14432637, 0.9016245, 0.09837545, 0.49773756, 0.5022624, 0.49748743, 0.50251
26, 0.45652175, 0.5434783, 0.4736842, 0.5263158, 9.99001E-4, 0.99900097]
Parents = [tampering, fire, Burglary, Earthquake]

Node: 5
Name = fire
Value1 = T
Value2 = F
Probability table = [0.01, 0.99]
Parents = []
```

Figure 16: The first part of merged network alarm1 and alarm2



```
Node: 6
Name = Burglary
Value1 = T
Value2 = F
Probability table = [0.3, 0.7, 0.02, 0.98]
Parents = [Night]

Node: 7
Name = tampering
Value1 = T
Value2 = F
Probability table = [0.02, 0.98]
Parents = []

Node: 8
Name = leaving
Value1 = T
Value2 = F
Probability table = [0.88, 0.12, 0.0, 1.0]
Parents = [alarm]
```

Figure 17: The second part of merged network alarm1 and alarm2