Assignment 2

190029087

15 Nov, 2019

1 Program Overview

The submission intends to implement the parts 1,2, and 3 for the assignment. The program can be run as:

2 Part 1

Figure 1 shows the single and multiply connected networks for modelling the traffic congestion problem. The domain for nodes of the network along with their symbols are mentioned in table 2.

Nodes and their values	
Node	Domain values
Weather	Sunny (SU), Overcast (OV), Rainy (RA), Snowy (SN)
Visibility sensor (VS)	Low (L), High (H)
Effective visibility (EV)	Low (L), High (H)
Peak time (PT)	True (T), False (F)
Day	Working (W), Holiday (H)
Car circulation (CC)	True (T), False (F)
Traffic flow (TF)	Low (L), High (H)
Camera output (CO)	Low (L), High (H)
Traffic congestion prediction (TCP)	True (T), False (F)
Emergency service alert (ESA)	True (T), False (F)

Both the networks rely on the following assumptions:

1. The peak time and the circulation of cars contribute to increase in traffic flow around the hospital area.

2. The visibility sensor being dependent upon the weather has probabilities P(H | SU) = 0.8, P(H | RA) = 0.3, P(H | OV) = 0.65 and P(H | SN) = 0.2 reflecting that visibility is worst on snowy days followed by rainy, overcast and sunny (the highest visibility).

3. The camera pictures the traffic flow with 95% accuracy and passes its output for prediction.

4. The effective visibility is the same as the output of the visibility sensor for 95% of the time and its output is also used for traffic congestion prediction.

5. The node 'traffic congestion prediction' is directly affected by the output of the camera as well as the effective visibility for checking traffic congestion prediction. A true (T) value for the node triggers the emergency service alert (ESA) with 90% probability.

.

In addition, BN2 introduces the following dependencies among the nodes:

1. Apart from the circulation of cars and the peak time, the weather is another factor directly affecting the traffic flow in the order: Rainy > Snowy > Overcast > Sunny. This can be backed by the fact that weather with bitter visibility introduce greater chances of road blockades (due to snow, flood, etc.) and hence increase the traffic flow on remaining roads.

2. Based on the fact whether it is a holiday or a working day, the peak time could drastically change. For working days, it is the same as the question suggests (i.e. 9/24 hours resulting in P(peak time = T) = 0.375) while I assume that the peak hour reduces to total of 2.4 hours on holidays thus resulting in P(peak time = T | day = H) = 0.1.

2.1 Querying the network

Each type of query is verified with two instances to provide further insight into the results.

Predictive: What are the chances of emergency service alert given its sunny weather, peak time and a working day?

For bn1: P(ESA | W = SU, PT = T, DAY = W) = 0.68

For bn2: P(ESA | W = SU, PT = T, DAY = W) = 0.47

We see that the probability for bn2 falls much below bn1. Given that bn2 employs multiple connections between 'weather' and 'ESA', the changes in weather are bound to have an effect on the visibility sensor as well as the traffic flow and thus the values are propagated down to ESA.

Diagnostic: If I observe that the effective visibility is high, and there is no emergency service alert, what is the likely weather?

For bn1: P(W | EV = H, ESA = F) = [P(W=SU): 0.42, P(W=OV): 0.16, P(W=RA): 0.32, P(W=SN): 0.09]

For bn2: P(W | EV = H, ESA = F) = [P(W=SU): 0.46, P(W=OV): 0.14, P(W=RA): 0.32, P(W=SN): 0.09]

The higher probability of the weather being sunny followed by overcast can be explained from the connection between weather and traffic flow nodes of 'bn2' which allows the values of ESA to be affected by weather through two paths. Consequently, the reverse effect on weather from ESA is greater for 'bn2'.

Profiling: How does the effective visibility affect its markov blanket?

From 1, we can see that a high effective visiblity contributes further to traffic congestion prediction being true in bn1. This can again be explained by the fact that the absence of direct relationship between weather and traffic flow in BN1 leads to stronger propagation of probabilities through the only available path. While the congestion prediction has identical values for low visibility, its parent node, *i.e.* camera output witnesses a dip in probability for high traffic in bn2 while remaining constant in bn1.

3 Part 2

Sample command for running the program from *A3src/src/*:

```
$ java A3main "multiplyConnected" "VE" "P(-pt|-day,-car)"
$ java A3main "singlyConnected" "VE" "P(-tcp|-day,-tf)"
```

List of arguments:

arg[0] network name to test. Three networks have been hard coded for initialization. Corresponding *xml* files can be found in *testCode/* directory.

arg[1] the inference algorithm - *VE* for variable elimination and *MC* for markov chain.

arg[2] the query for the corresponding network. Literals in the query are node names in lowercases. A '-' sign before a literal indicates a negative value. Also, the 'high' and 'low' values are mapped to 'true' and 'false' wherever applicable.

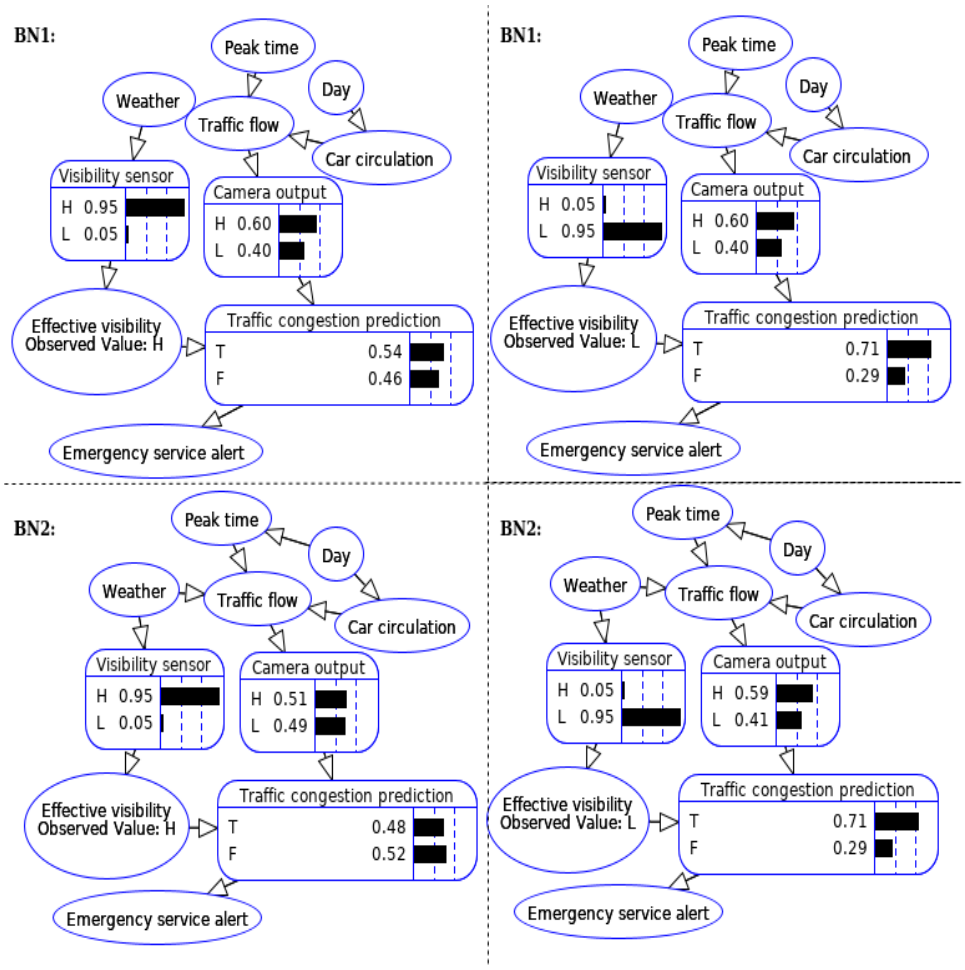

Figure 1: Figure profiling the effects of effective visibility.

3.1 Design

The object-oriented design for construction of bayesian network includes the following common modules that facilitate the initialization of the network based on the information provided by the user:

Class	Functionality
Bayesian Network	Class defining methods such as addNode() for adding a variable to the network and parseObservation() for parsing the user-inputs.
RandomVariable	Class for actual mapping of a variable's name to a node in the network along with maintaining its children, parents and set of probabilities for possible values.
DomainValue	Class mapping an instance of RandomVariable with its unique name thus easing the search for a variable.
Observation	Class representing a variable and its observed value.
ObservationCondition	Class maintaining a list of all possible observations for a given variable.
CustomError	A manual exception class that is raised when the network or a query shows anomalous behaviour.

The implementation of variable elimination algorithm follows the following steps:

Step 1 Using the observation and the queried node, all the nodes of the network that fall in the path between them are extracted and the irrelevant nodes are pruned.

Step 2 The algorithm then loops for each of these nodes starting from the topologically last node in the path and performs following steps:

(a) The factor for each node is initialized with its original conditional probability table.

(b) For each node that is neither a target nor an evidence, its factor is multiplied with the factor from the previous node to form a new factor. The variable is then removed from the list of variables comprising the factor for the current node in the loop. The elimination is followed by marginalization of probabilities of the variable with remaining variables of the factor.

(c) The last step of each loop then involves replacing the previous list of factors with the newly formed factor so that the size of the list of factors is always limited to 2 upon processing of next node. This reduces the computational burden at each step.

Step 3 After having processed all nodes, the remaining factors are joined using point-wise product and normalized to get the sum of resulting probabilities being 1 for all domain values of the queried variable. The probability for queried value is then returned.

3.2 Implementation

The method decomposition followed by the implementation for building of bayesian networks is loosely based on Russell Norvig's AIMA code repository[1]. The main class *A3main.java* initializes the network to be queried upon, parses the user input to extract the evidence and the queried node, and feeds it to the *processQuery()* method where the steps in section 3.1 are executed. The class *Factor()* additionally contains the constructor code for creating factors given a RandomVariable v and its observation as well as the modules *join()* for dot product, *eliminate()* for removal of a variable and marginalization of its probabilities and *normalise()* for normalization of the final probabilities of the queried node.

4 Evaluation

The code has been evaluated on three separate networks: one as shown in figure 5, other being a multiply connected version of bn1, and the last one as provided with the assignment.

4.1 Evaluating network BNpart2

Command:

```
$ java A3main "BNpart2" "VE" "P(leaving|-fire)"
```

Figure 2 shows the steps involved in answering the above query which is parsed as P(LEAVING=T|FIRE=F). The final answer is 0.748 which was verified using AISpace tool.

4.2 Evaluating network singlyConnected

Command:

```
$ java A3main "singlyConnected" "VE" "P(-esa|car)"
```

Above predictive query asks -*i.e.* what are the chances of the emergency service alert given that car circulation is high? It is parsed as P(ESA = F | CAR = T). Final result is 0.3612 which can be verified using AISpace (figure 5).

4.3 Evaluating network multiplyConnected

Command:

```
$ java A3main "multiplyConnected" "VE" "P(-co|-day)"
```

Above query asks the chances of camera output being low given a holiday. Final result is 0.689 which can be verified using AISpace (figure 4).

[1]https://github.com/aimacode/aima-java/blob/AIMA3e/aima-core/src/main/java/aima/core/probability/bayes/BayesianNetwork.java

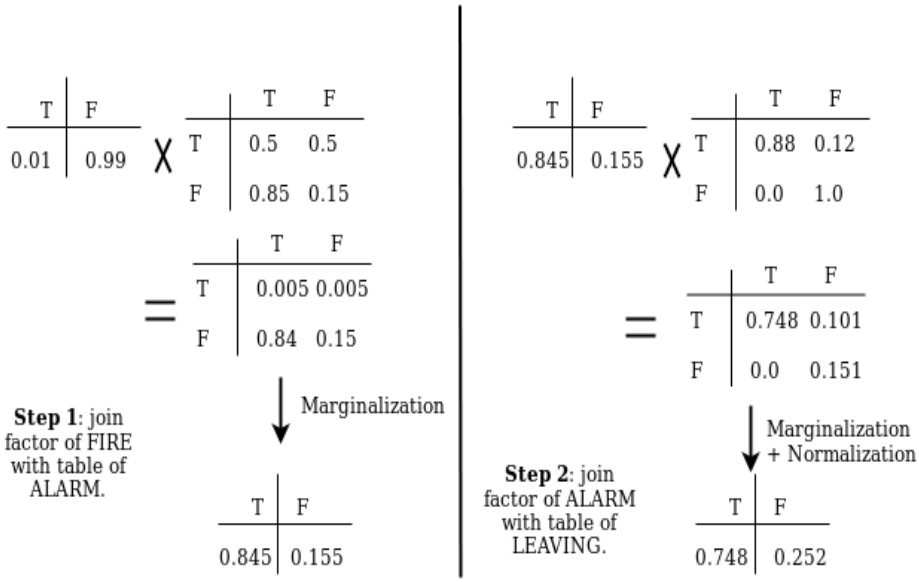

Figure 2: Figure showing the factorization and joining steps for the query P(leaving|-fire).

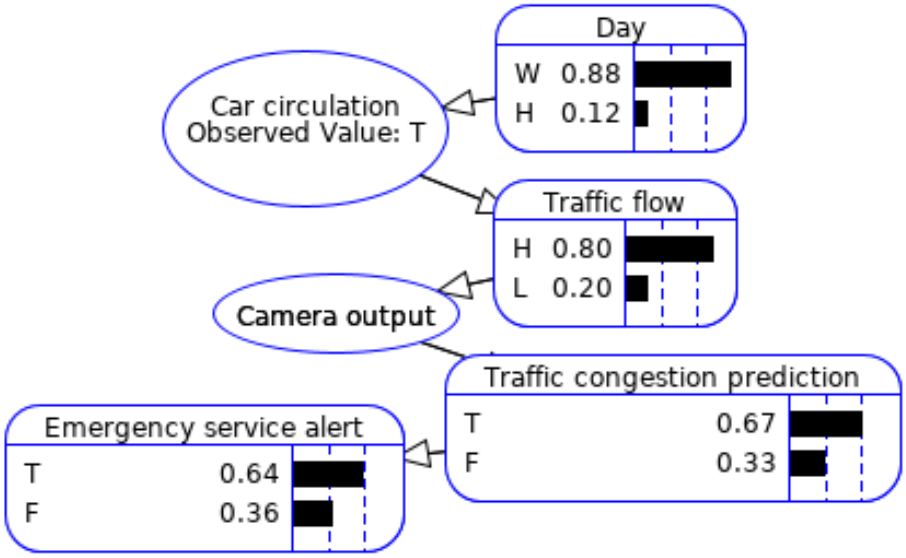

Figure 3: Figure showing the verification of query using AISpace.

5 Part 3

5.1 Defining own problem

The problem formulation models the launch of a satellite and its flight to ensure its chances of landing on the surface of Mars. The system can be found within

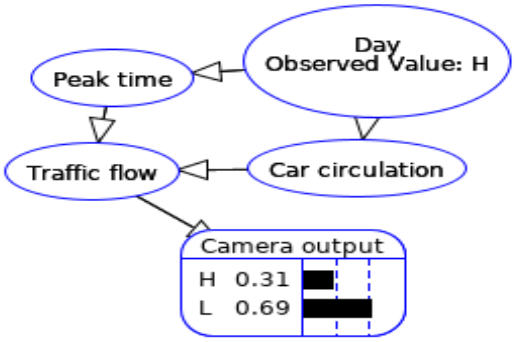

Figure 4: Figure showing the verification of query using AISpace.

the file **advanced.xml** and relies on the following factors:

1. As the rocket carrying the satellite reaches an altitude of around 20 kms from the sea level, its solid rock motors are jettisoned. While 45% cases of jettisoning are successful in first-go, 35% of them might require repeated trials until all the motors are dropped and 20% of the cases result in disasters where the motors might slide down the side of the rocket's core instead of cleanly departing.[2] The resulting collision can in turn cause low, medium or severe failure of sensors on the satellite depending upon the severity of the collision.

2. Depending upon the sensors damaged during the jettisoning, the satellite might not be able to communicate with the control team back on Earth or sense its altitude after having entered the Martian atmosphere. Consequently, the parachute deployment might be a success or failure.

3. The Martian weather at the moment of landing could be sunny, cloudy or full of dust storms. While the sunny weather favors a successful landing, the storms worsen its chances the most.

4. The failed sensors can directly affect the satellite's capacity of examining its altitude and thus interfere with the process of parachute deployment. Low levels of failure can prevent the deployment by 35%, medium failures by 60% while critical failures by 82%.

5. Finally, a successful landing depends upon the severity of sensor failure, the success of parachute deployment as well as the weather at the time and location of landing.

The queries performed are:

[2]https://www.nasaspaceflight.com/2018/11/soyuz-ms-10-abort-sensor-failure-booster-separation/

Predictive: If weather = stormy and sensor failure = critical then, the chances of landing must be worse. The result of query: P(landing successful = T) = 0.0254 thus agrees with the assumptions.

Diagnostic: If the landing was successful yet the sensor failure 'critical', then querying booster separation gives P(outcome = disaster) = 0.6567 and P(outcome = prolonged) = 0.209 as top two candidates. Thus the jettisoning must have caused serious collisions with the rocket carrying the satellite.

Profiling: How does sensor failure affect its markov blanket? We can see that as

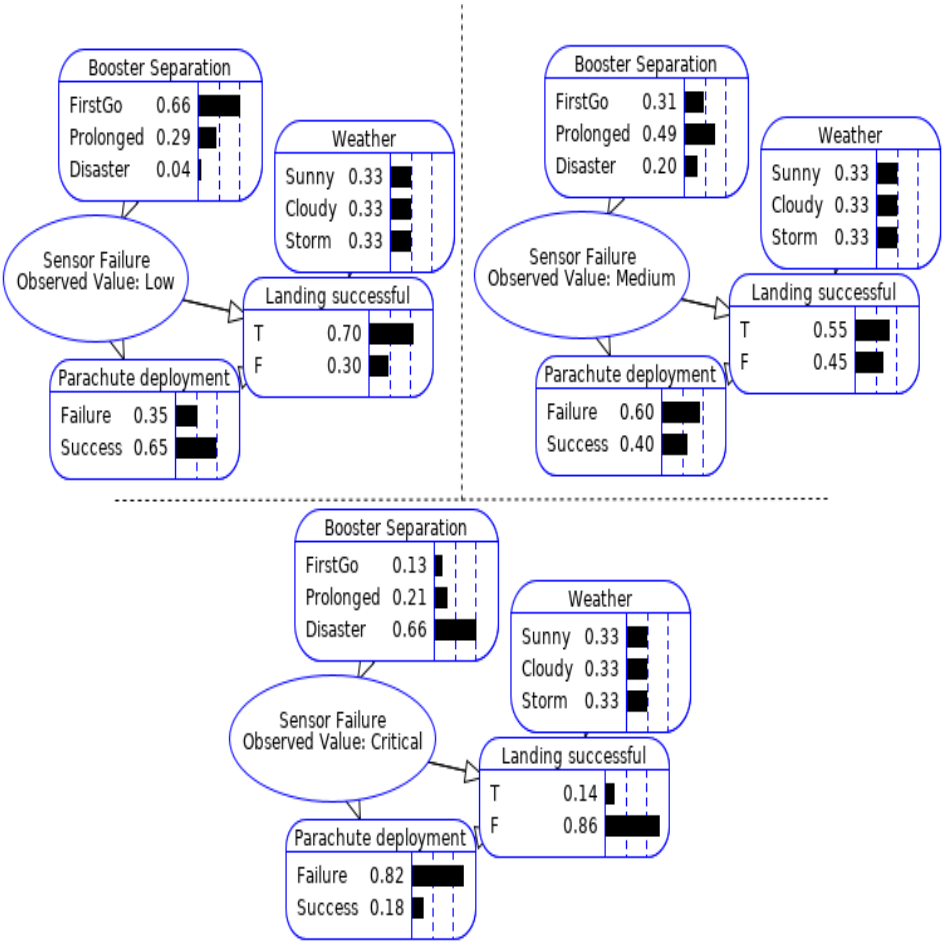

Figure 5: Figure profiling the effect of sensor failure.

the severity of sensor failure increase from low to medium and then critical, its parent, i.e. booster separation phase, can be diagnosed to shift

towards prolonged and finally disaster. Similarly, the chances of failure in parachute deployment failure and landing also skyrocket. In contrast, this does not infer anything about the weather since the probabilities for weather remain constant throughout.

5.2 Additional algorithm for inference: Markov chain simulation through Gibbs sampling

The program implements Markov chain Monte Carlo algorithm as an advanced inference-making method using Gibbs sampling. The algorithm can be described in following steps:

Step 1 The algorithm maintains a sample state of the network which is a list containing the name of all nodes and their current values, and is initialized with a two-step process:

(a) All the nodes in the network forming the evidence of the query are added to the sample state with their values as those in the evidence.

(b) While adding the rest of the nodes that are neither evidence nor the node being queried, we simply assign a random value from its domain.

Step 2 The next step involves iterating through N number of loops (N=1500 for our purpose) and randomly sampling one of the non-evidence variables X_i. We then draw a new sample value x_i for X_i for the non-evidence variable from its domain conditioned on its Markov blanket. The step handled by the *getSample()* method assumes that the probability of a variable given its Markov blanket (mb) is proportional to the probability of the variable given its parents times the probability of each child given the child's parents, i.e.

$$P(\overline{x_i}|mb(X_i)) = \alpha P(\overline{x_i}|parents(X_i)) * \prod_{Y_j \in children(X_i)} P(y_j|parents(Y_j)) \tag{1}$$

For each value in the domain of the randomly sampled variable, the method performs following steps:

(a) Given the value of the non-evidence variable, its probability is calculated conditioned on its parents' values in the current sample state.

(b) The above probability is then multiplied with the probability of each of its child conditioned on the values of the child's parents in the current sample state.

The resulting probability is then stored for each value of the variable and is normalized to make their sum equal to 1. Post normalization, the value with a higher probability is returned as the resulting x_i for updating X_i in the sample state. Lastly, a counter is incremented if the value of

the queried variable at the current state is equal to the value asked by the user's input. This assumes that value of the counter in the long-run fraction of time spent in each state will eventually reflect a state of *dynamic equilibrium* which is directly proportional to its posterior probability.

Step 3 The normalized value of counter is returned after N iterations.

5.2.1 Query outputs

The same queries are used validation of Markov Chain as in variable elimination. Note that the results here might fluctuate with each run of the program. I have therefore, reported the results in the range they mostly occurred after five runs.

5.3 Evaluating network BNpart2

Command:

```
$ java A3main "BNpart2" "MC" "P(leaving|-fire)"
```

Output: mostly in the range [0.695, 0.850] which resembles the results of AISpace being 0.748.

5.4 Evaluating network singlyConnected

Command:

```
$ java A3main "singlyConnected" "MC" "P(-esa|car)"
```

Output: mostly in the range [0.274, 0.390] which again resembles to AISpace's result of 0.360.

5.5 Evaluating network multiplyConnected

Command:

```
$ java A3main "multiplyConnected" "MC" "P(-co|-day)"
```

Output: [0.650, 0.780] which is close to AISpace's result of 0.690.