1. Introduction

Assignment Number: A3
Date: 30/03/2022
Student ID: 210017984
Word Count: 1580 excluding tables, figures, captions.

1.1 Parts Implemented

Part	Status
Part 1	Attempted, fully working.
Part 2	Attempted, fully working.
Part 3	Attempted, fully working
Part 4	Attempted, fully working
Part 5	Attempted, 1. Implemented own Gibbs Sampling – not tested thoroughly. 2. Details flag – minor extension used for evaluation, used to measure output times and comparisons made.

Table 1: Parts implemented.

1.2 Compiling & Running Instruction

1. Navigate to the **src** directory included in the AI-Inference folder:

```
cd AI–Inference/src/
```

2. From within src, **compile** the program:

```
javac *.java
```

3. Run it:

```
java A3main <Pn> <NID> <details|greedy>
```

The first argument specifies the part of the practical (P2, P3, P4, and P5 available) and NID indicates the network ID. The available networks are BNA, BNB, BNC, and CNX. If a valid <Pn> and <NID> are given, then the program asks for a Query, Order (P2, P3), and Evidence (P3, P4). P5 is the limited functionality gibbs sampling. The <details> flag specifies that the user wishes to run the system recording average performance details. The <greedy> flag specifies to use the Greedy Minimum Edges Search for P4.

Running Example:
Run the program using Part 3 on the BNC network. Given query = Z:T, order = P,U,R,S,Q,V, and evidence = R:T U:T.

```
java A3main P3 BNC

Query:
  Z:T
Order:
  P,U,R,S,Q,V
Evidence:
  R:T U:T
```

2. Design & Implementation

2.1 PEAS (Performance, Environment, Actuators, Sensors)

PEAS Model				
	Performance Measure	**Environment**	**Actuator**	**Sensor**
P2 P3	1. Correct probability 2. Number of operations performed.	1. The Bayesian network given and its nodes. This includes nodes children and parents. 2. The environment is **dynamic** and **unpredictable** as CPTs are continually updated as the agent makes new inferences.	1. Variable elimination process following the order. 2. Join and marginalisation operations. 3. Normalization operations.	1. The CPT tables and the order given.
P4				1. The CPT tables, the order, and evidence given.
				1. The CPT tables and evidence given. 2. The order calculated using the Maximum Cardinality Search.

Table 2: PEAS model for agents.

2.2 Classes Infrastructure

The developed system involves 6 classes (excluding main), which are BayesianNetwork, Node, Edge, CPT, VariableElimination, Ordering, and GibbsSampling.

Creation of a Bayesian Network

Figure 1 shows the interaction between the four classes involved in the creation of a Bayesian Network. The BayesianNetwork class contains multiple nodes (Node class), where each node represents a network's variable. Each pair of nodes are connected between them using edges (Edge class). Each variable/node contains a list containing its parents and children, as well as a conditional probability table (CPT class) that displays the conditional probabilities of a single variable with respect to its parents (i.e., the probability of each possible value of one variable if we know the values taken on by the other variables).

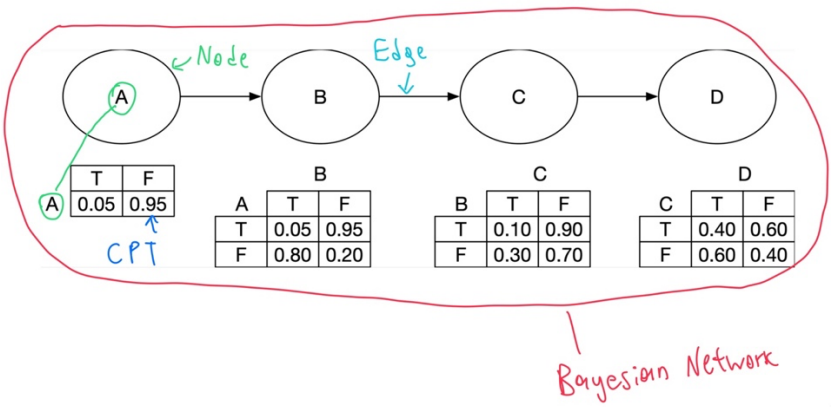


Figure 1: The bayesian network structure - which part goes to which class.

Table 3 briefly describes the most important methods of each class in the Bayesian Network infrastructure.

Most Important Methods - classes used to create a Bayesian Network		
Class	**Method**	**Purpose**
BayesianNetwork	addNode	Creates and adds a new node (Node object) to the Bayesian Network.
	addEdge	Create a new Edge object that connects two given nodes.
Edge	Edge (constructor)	Creates a new edge connection between two nodes. The *first* node is added as apparent to the *second*, and the *second* is added as a child to the first.
Node	addChild & addParent	Adds a new child to the node (*addChild*) or a new parent (*addParent).*
	getAllNeighbours	Returns all the neighbours (parents, childrens, and co-parents) of a node.
CPT	addCPTvalues	Add given probability values for each combination in the CPT table.
	populateMap	Populates a truth combination - value map. Very important for adding the correct values to the correct CPT variable combinations. A map data structure was chosen to be able to easily get and manipulate the value of a truth value combination. The map has the all the truth combinations stored each stored an ArrayList as keys and a probability value stored as a double.
	setToZero	When evidence factor is equal to true then set all values of the CPT in EvFactor where E = F to zero, and vice versa.

Table 3: Most important methods in classes involving Bayesian Network (BayesianNetwork, Edge, Node, CPT)..

VariableElimination Class

The VariableElimination class contains all the methods required to perform variable elimination, an exact inference algorithm that can be used in probabilistic graphical models for maximum a posteriori inference and estimation of conditional or marginal distributions over a subset of variables. The key method inside this class is called *runVE* and is used to drive the operation of variable elimination. Other important functions are *join* and *marginalise*. The class most important functions are detailed in Table 1.

Most Important Methods - VariableElimination Class	
Method	**Purpose**
runVE	Run the variable elimination algorithm given the value being looked for and the evidence.
pruneIrrelevantVariables	Prunes all the irrelevant variables according to the task. If the evidence flag is true (P3, P4, P5), then it deletes ancestors of the evidence and the queried variable. Otherwise (P2), it just removes every variable that is not an ancestor of the queried variable.
createSetFactors	Create a set of factors for every node in the Bayesian network that is included in the order or is the queried variable.

JoinMarginalise	The join marginalise operation used to eliminate factors.

Table 4: The most important methods in the variableElimination class.

The *joinMarginalise* method is the most important method in this project as it uses the *join* and *marginalise* functions to sum out all the factors that contain a given label (the one currently in order) and then marginalise the label out of the resulting CPT table.

The join function (Figure 2), joins two CPTs together by multiplying their corresponding probabilities where the common variable matches. This results in a new CPT being created that contains the join probabilities of the two factos.

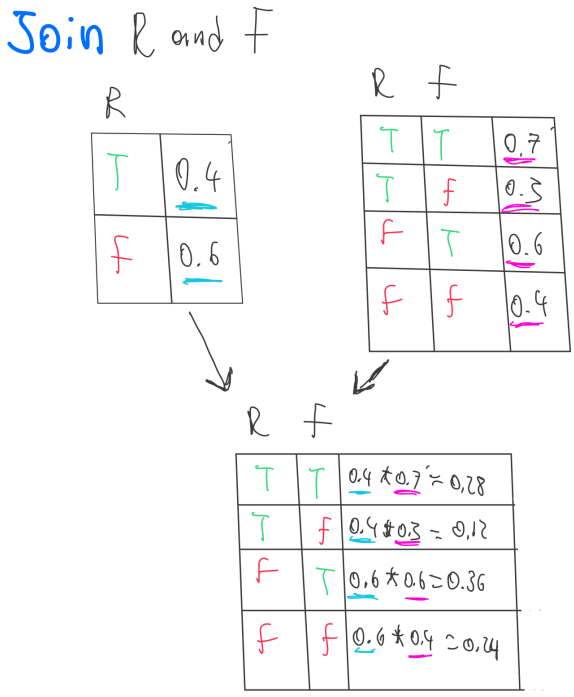


Figure 2: Join operation between two CPTs.

The next step in the process is to marginalise/remove the label out of the new resulting table. For example in Figure 3, R is marginalised by adding the CPT probabilities when F is true (R's true and False when F=true), and vice versa.

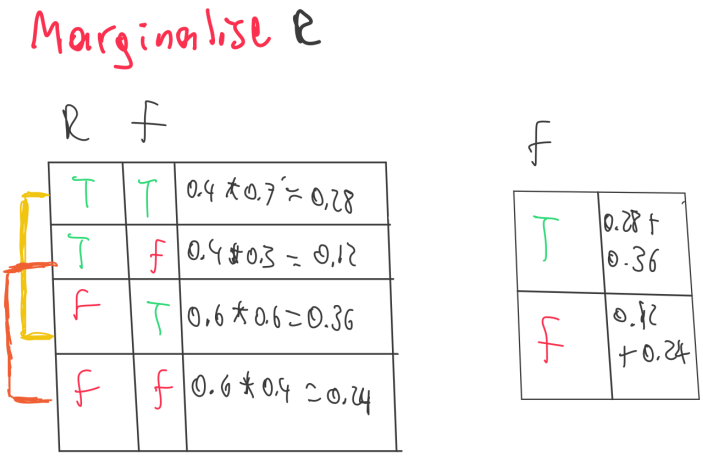


Figure 3: Marginalisation operation.

Ordering Class

The ordering class includes all the methods required by the two variable elimination ordering algorithms, maximum cardinality search and greedy minimum edges.

The maximumCardinalitySearch algorith iterates through all the network's nodes and each time adds to the order the label with the maximum marked neighbours that is currently unmarked. The resulting order is reversed, and the queried variable is removed from it. We are using the network as an **undirected** graph as we are working with each nodes neighbours (children, parents, co-parents), not only with children or parents.

The Greedy Minimum Edges algorithm was also attempted but it results in a java memory heap error, I believe due to my very limited computer resources (forced to use old computer since my new malfunctioned last week).

GibbsSampling Class (Extension)

Gibbs sampling is used to obtain a sequence of approximations for observations from a specified multivariate probability distribution when exact inference is very difficult to calculate. I tried to implement gibbs sampling by following the pseudo code (Figure 4) in the AI a modern approach book[1]. To do that I created a new class called gibbsSampling which contains a function called *gibbsAsk* that takes the number of samples and the truth value we are looking for. The algorithm first gets all the non-evidence nodes and assigns them a value (binary). Then it iteratively samples each of the nodes in the non-evidence set from their **full conditional** probability using the node's previous value. The conditional full probabilities or both true and false are calculated and then normalized. Then a normal value between 0 and 1 is generated. The normalized **true** probability is used as a threshold and if the generated value is between 0 and the normalized true probability, then that node's value gets updated to 1, otherwise to 0. The idea is that if there are enough samples after we count them for the specified variable we should be able to estimate really closely the probability.

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X \mid \mathbf{e})$
  local variables: **C**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initialized from **e**

  initialize **x** with random values for the variables in **Z**
  for $k = 1$ **to** N **do**
    choose any variable Z_i from **Z** according to any distribution $\rho(i)$
    set the value of Z_i in **x** by sampling from $\mathbf{P}(Z_i \mid mb(Z_i))$
    $\mathbf{C}[j] \leftarrow \mathbf{C}[j] + 1$ where x_j is the value of X in **x**
  return NORMALIZE(**C**)

Figure 4: Gibbs Ask pseudo code.

Unfortunately, I was not able to test the develop algorithm extensively as my computer is broke recently and I am forced to use my old computer which is struggling to generate more than 1000 samples, which of course are not enough for this process. With the limited testing I did, I believe there must be a mistake in the calculation process as the values I got where mostly not close to the exact inference result.

2.3 CNX Network

Figure 2 shows the Bayesian system designed for predicting the risk of digital security attacks for a company network CNX. The alert system was designed following the motto: **causes precede effect** and can help the CNX digital security officers to **predict** potential computer attacks. To develop the network, I used the probabilities given in the specification as well as some further logical assumptions I made to fill out the remaining CPT values, which are listed below.

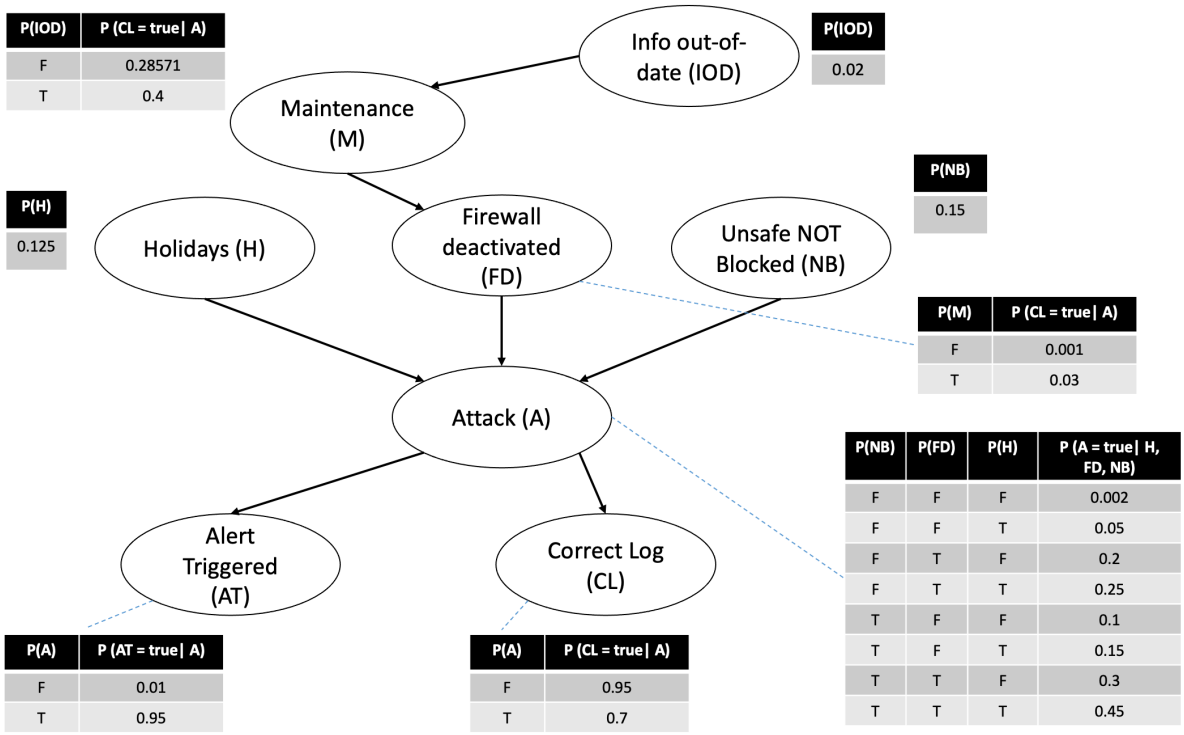


Figure 5: Bayesian Network developed for CNX.

Variable	Assumptions
Maintenance (M)	The company is very cautious and as such system maintenance (M) occurs twice a week (2/7) ➔ P(M) = 0.28571 (5 d.p.). If the information is out-of-date (IOD), it has a direct influence on the planned Maintenance (M), increasing the probability of Maintenance to 40% to reflect the uncertainty - increased likelihood that there is an upcoming planned maintenance that we are unaware of.
Attack (A)	The risk of an attack (A=True) is the highest when the firewall is deactivated (FD=True), an unsafe website is not blocked (NB=True), and it is a holiday (H=True), in that case, I estimate a 45 percent chance of an attack. This value was chosen because it is quite likely there might have been an attack given the circumstances, but not certain as computer attacks are realistically not that common – even if there are enabling vulnerabilities/opportunities. The probability of an attack is the lowest in the exact opposite scenario - (H=False, FD=False, NB=False), assuming a value of 0.2%. Regardless of how unlikely it is, an attacker may have managed to bypass our firewall and security layers when the firewall was activated, unsafe websites were blocked, and was not a holiday.

	The variable with the greatest influence on an attack is the firewall deactivated (FD), followed by unsafe not blocked (NB), and finally by holidays (H). As a result, when given FD=True, NB=True, and H=False, I assume the second highest probability, 30%. The rest of the probabilities were filled in following this logic.
Alert Triggered (AT)	There is a 1% chance that the alert will be triggered as a False Positive (i.e., when no attack is predicted - A=False), as alert systems often have oversensitive settings for precautionary purposes.
Correct Log (CL)	The probability of a correct log when an attack was not predicted (A=False) is 95% as it is better trained in classifying normal situations than malicious ones due to having a lot more examples. The company is aware of this vulnerability and working towards fixing it.

Table 5: CNX network assumptions.

The network was implemented in the A3main class using the Bayesian Network infrastructure explained in 2.1. The testing networks were also implemented there.

3. Testing

3.1 Tests Provided

BNA

Part	Input	Output	Expected	Correct
A	java A3main P1 BNA	Prints the network	Prints the network	**YES**
B	java A3main P2 BNA Query: D:T Order: A,B,C	0.57050	0.57050	**YES**
	java A3main P2 BNA D:T B,C,A	0.57050	0.57050	**YES**
C	java A3main P3 BNA Query: D:T Order: A,B,C Evidence: A:T	0.54200	0.54200	**YES**
	java A3main P3 BNA D:T A,B,C A:T B:T	0.58000	0.58000	**YES**
	java A3main P3 BNA A:T D,B,C C:T	0.09831	0.09831	**YES**
	java A3main P3 BNA D:T A,B,C B:T	0.58000	0.58000	**YES**
	java A3main P3 BNA D:T A,B,C A:F B:T	0.58000	0.58000	**YES**
COMMENT:	All output is correct.			

Table 6: BNA tests expected and actual.

BNB

Part	Input	Output	Expected	Correct
A	java A3main P1 BNB	Prints the network	Prints the network	**YES**
B	java A3main P2 BNB Query: N:T Order: J,L,K,M,O	0.39864	0.39864	**YES**
	java A3main P2 BNB M:T J,L,K,N,O	0.49660	0.49660	**YES**
C	java A3main P3 BNB	0.54385	0.54385	**YES**

	Query: K:T Order: O,J,M,L,N Evidence: O:T			
	java A3main P3 BNB J:T O,M,L,K,N O:T	0.04233	0.04233	**YES**
	java A3main P3 BNB N:T J,L,K,M,O J:T	0.43360	0.43360	**YES**
	java A3main P3 BNB N:T J,L,K,M,O J:T L:T	0.42400	0.42400	**YES**
	java A3main P3 BNB N:T J,L,K,M,O J:T L:F	0.45600	0.45600	**YES**
COMMENT:	All output is correct.			

Table 7: BNB tests expected and actual.

BNC

Part	Input	Output	Expected	Correct
A	java A3main P1 BNC	Prints the network	Prints the network	**YES**
B	java A3main P2 BNC Query: U:T Order: P,R,Z,S,Q,V	0.42755	0.42755	**YES**
	java A3main P2 BNC S:T P,U,R,Z,Q,V	0.49660	0.49660	**YES**
C	java A3main P3 BNC Query: Z:T Order: P,U,R,S,Q,V Evidence: R:T U:T	0.43368	0.43368	**YES**
	java A3main P3 BNC P:T U,R,Z,S,Q,V Z:T	0.05509	0.05509	**YES**
	java A3main P3 BNC Q:T P,U,R,Z,S,V Z:T S:T	0.92141	0.92141	**YES**

	java A3main P3 BNC U:T P,R,Z,S,Q,V Z:T Q:T R:T	0.34204	0.34204	**YES**
COMMENT:	All output is correct.			

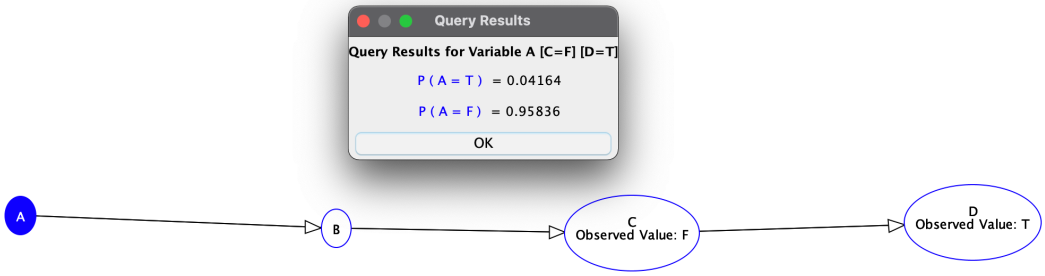
Table 8: BNC tests expected and actual.

3.2 Additional Tests

The AISpace tool was used to further test the exact inference values returned by the variable elimination method on additional test cases. All the results were calculated correctly. I ran them using the best order (P4).

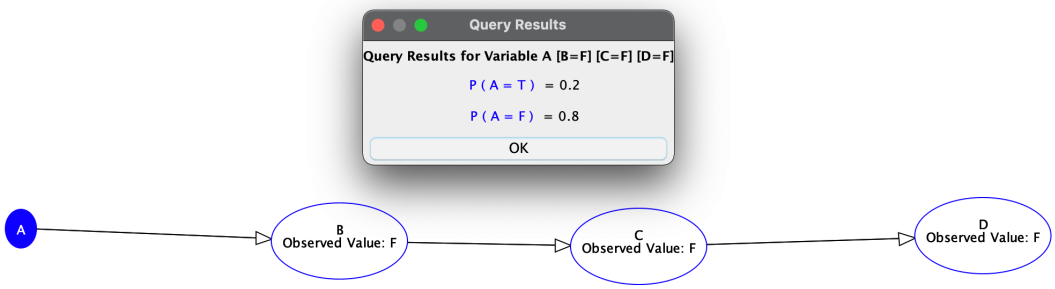
BNA

1. P(A | C=F, D=T) - Correct




```
Query:
A:T
Evidence:
C:F D:T
[D, C, B]
0.04164
```

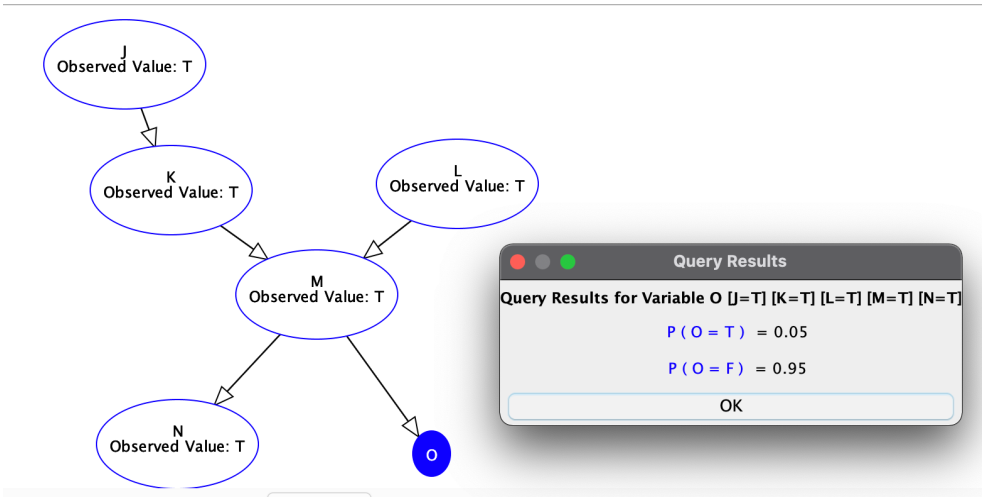
2. P(A | B=F, C=F, D=F) - Correct




```
Query:
A:T
Evidence:
B:F C:F D:F
[D, C, B]
0.20000
```

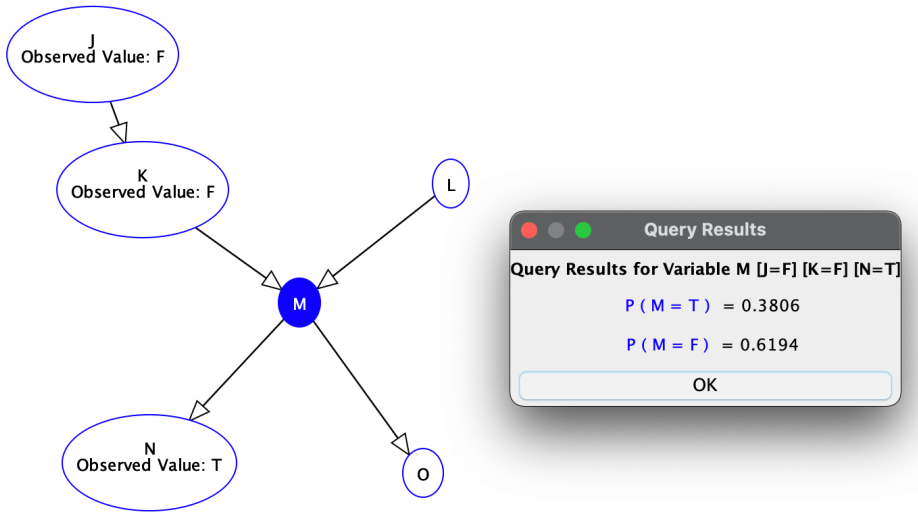
BNB

1. P(O | J=T, K=T, L=T,M=T, N=T)- Correct




```
Query:
O:T
Evidence:
J:T K:T L:T M:T N:T
[J, N, K, L, M]
0.05000
```

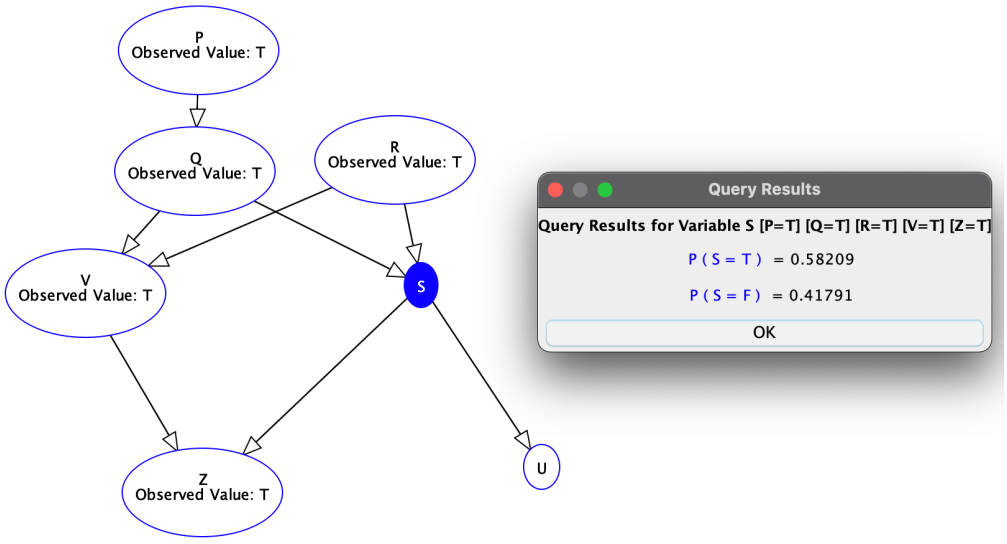
2. P(M | J=F, K=F, N=T)- Correct




```
Query:
M:T
Evidence:
J:F K:F N:T
[O, J, N, K, L]
0.38060
```

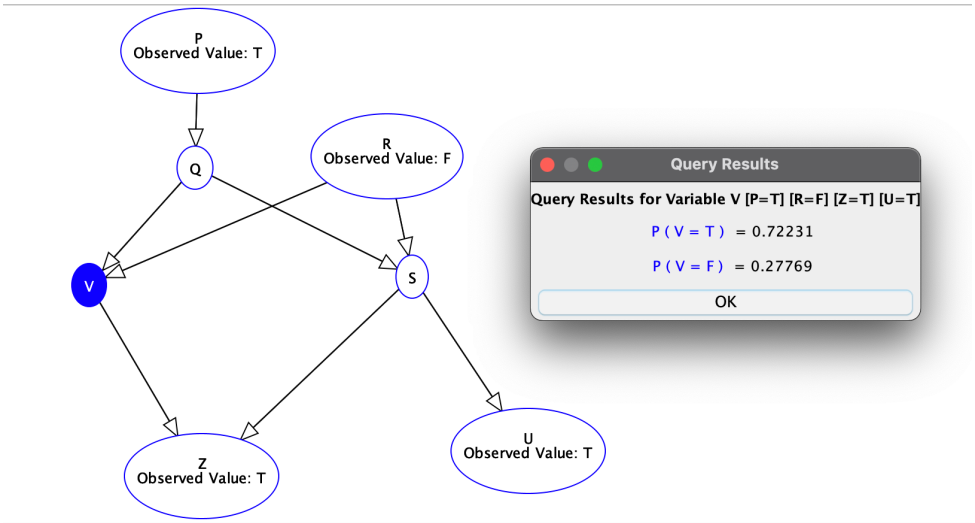
BNC

1. P(S | P=T, Q=T, R=T, V=T, Z=T)- Correct




```
Query:
S:T
Evidence:
P:T Q:T R:T V:T Z:T
[U, P, Z, V, Q, R]
0.58209
```

2. P(V | P=T, R=F, Z=T, U=T)- Correct

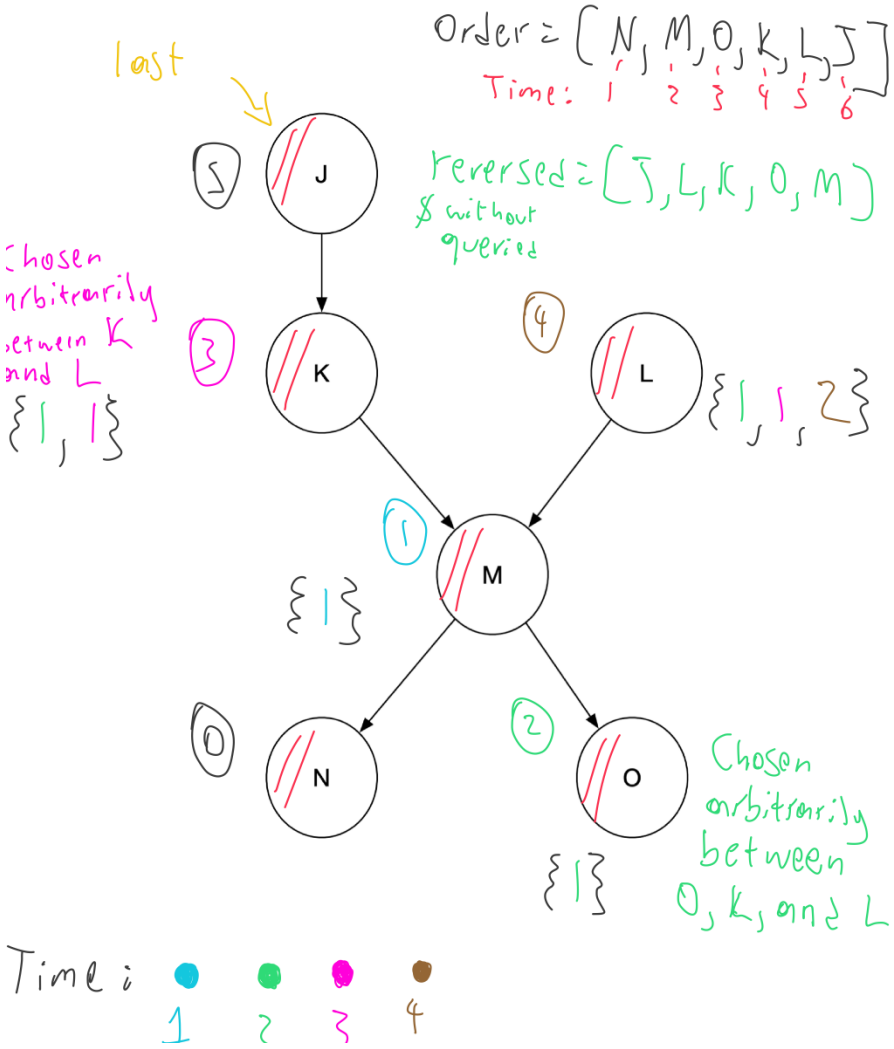



```
Query:
V:T
Evidence:
P:T R:F Z:T U:T
[U, P, Z, S, Q, R]
0.72231
```

3.3 Ordering By Hand

BNB – Query N:T

The graph here shows the sequence of steps that the nodes should be marked and added to the queue. Step 0 and 5 are in black colours as no choice had to be made.




```
(base) Rafaels-MBP:src rafaelkoll$ java A3main P4 BNB
Query:
N:T
marked: [Node{N}]
order: []
Step 0 marked: [Node{N}]
Step 0 order: []
Step 1 marked: [Node{N}, Node{M}]
Step 1 order: [M]
Step 2 marked: [Node{N}, Node{M}, Node{O}]
Step 2 order: [M, O]
Step 3 marked: [Node{N}, Node{M}, Node{O}, Node{K}]
Step 3 order: [M, O, K]
Step 4 marked: [Node{N}, Node{M}, Node{O}, Node{K}, Node{L}]
Step 4 order: [M, O, K, L]
Step 5 marked: [Node{N}, Node{M}, Node{O}, Node{K}, Node{L}, Node{J}]
Step 5 order: [M, O, K, L, J]
Evidence:
J:T
[J, L, K, O, M]
```

Print statements were added for demonstrating purposes. As evident the ordering algorithm implemented behaves correctly.

Gibbs Sampling

I believe that gibbs sampling method implemented does not work as expected, returning wrong results. Given more time and more capable computer, I would have debugged this extension extensively to fix the problem. I

4. Evaluation

Query (Task 2)

Table 9 shows how different ordering affects the algorithm's performance in the same query (averaged over 20 runs). Given no evidence, the query calculates the probability that the alert will be triggered (AT=True).

Elimination Ordering Comparison - Average of 20 runs						
Query	**Node Elimination Order**	**Order Reasoning**	**Running time (µs)**	**JoinMarginalise Operations**	**Truth Values Calculated**	**Result**
AT:T	IOD,M,H,FD,NB,A,CL	Top-to-Bottom - Ascending	2098 µs	69	690	0.03313
AT:T	CL,A,NB,FD,H,M,IOD	Bottom-to-Top - Descending	4376 µs	69	1449	0.03313
AT:T	M,NB,FD,H,A,CL,IOD	Random	2627 µs	69	1035	0.03313

Table 9: Elimination ordering Comparison.

As evident, the order of node elimination has no effect on the probability (result) returned or the number of joinMarginalise operations performed. The number of truth values calculated, on the other hand, is significantly correlated with the elimination ordering, as when a poor ordering is given, the algorithm must compute larger factors, which contain more truth values. In this example, when ascending ordering is used, the average running time and number of truth values calculated are significantly less than the second best (random), and twice as good as descending ordering. As shown in Figure 6, there is a strong correlation between running time and calculated truth values. The result is reasonable because the queried node is at the bottom, implying that a top to bottom elimination ordering is most appropriate for exact inference.

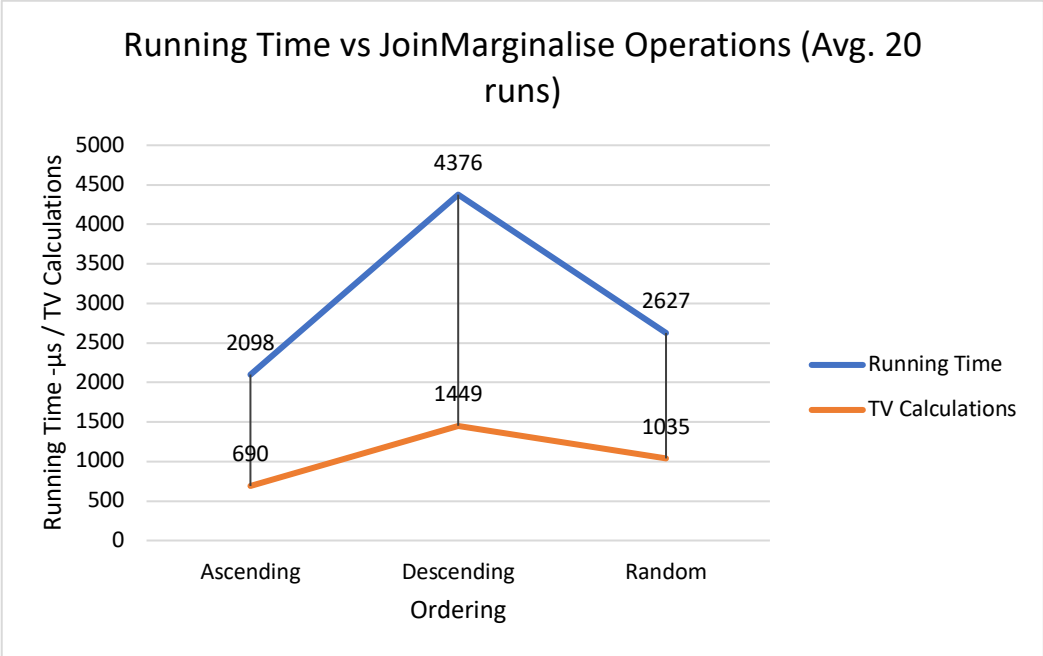


Figure 6: Running time vs joinMarginalise operations.

Predictive vs Diagnostic (Task 3)

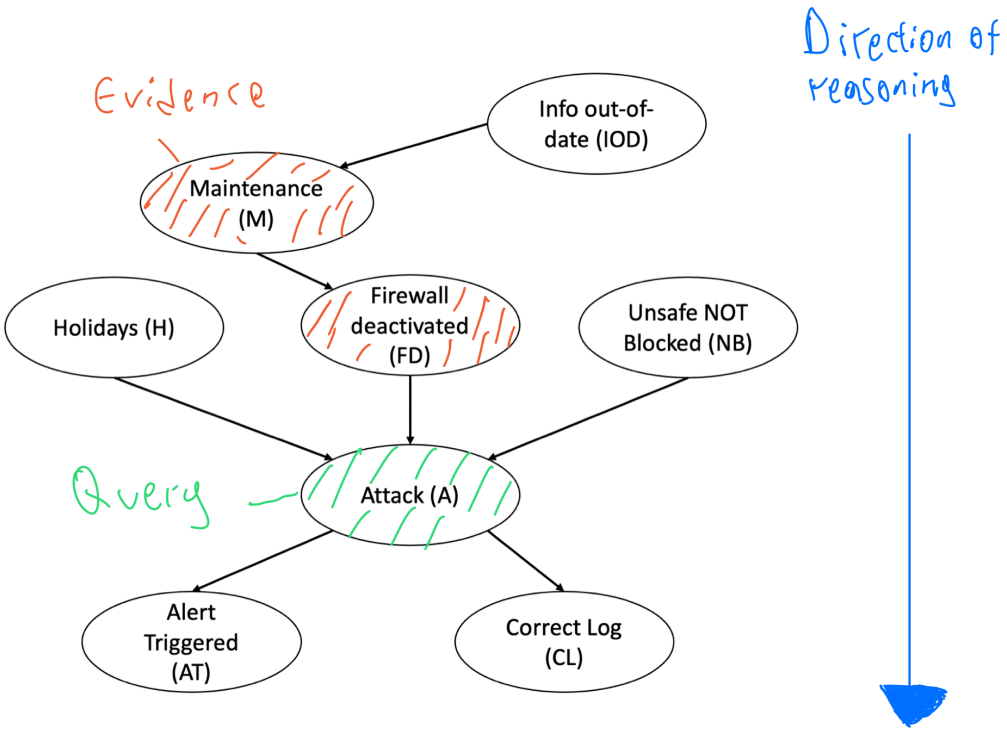


Figure 7: Predictive query.

Figure 7 shows a **predictive** query where we are predicting the probability of an attack (A=True) given that there is planned maintenance (M=True) and the Firewall Deactivated varies (tested both cases). In predictive queries we are trying to predict future observations given past observations. In Bayesian statistics this predictive probability can always be calculated exactly (given valid CPTs and BN structure).

Predicting Potential Attack Given Evidence - Predictive			
Query	**Node Elimination Order**	**Evidence**	**Result**
A:T	IOD,M,H,FD,NB,AT,CL	M:T FD:T	0.22313
A:T	IOD,M,H,FD,NB,AT,CL	M:T FD:F	0.02274

Table 10: prediction pottential attack given evidence – Predictive query.

The first row shows that the probability of predicting an attack given that there is a planned maintenance (M=T) and the firewall is deactivated (FD=T) is 22.31%. In the second row we observe that when firewall is not deactivated (FD=F), the probability for an attack drops significantly to 2.27%. This indeed makes sense as even if there is planned maintenance, the firewall is **very rarely deactivated**. However, if it is deactivated then it significantly increases the possibility of an attack as explained in the assumptions (See 2.3). The system's predictive abilities can be used by the company to predict when the network is susceptible and thus issue a high alert message to the officers.

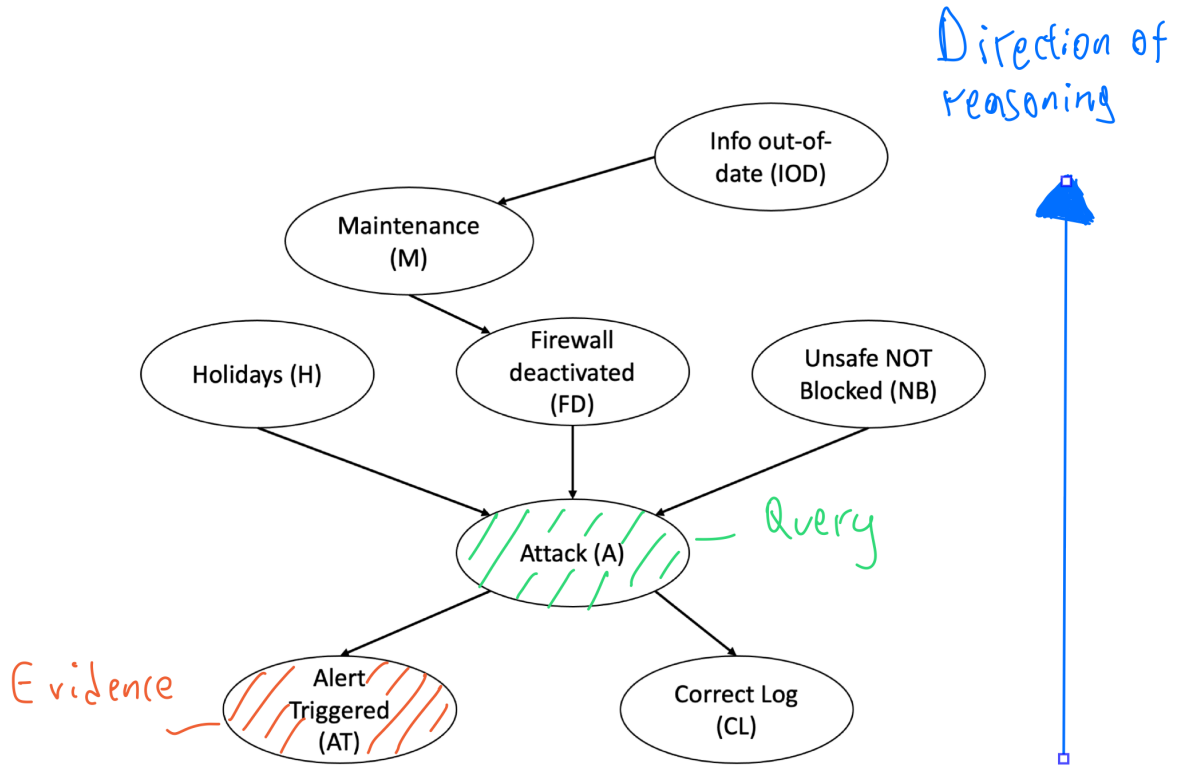


Figure 8: Diagnostic query.

In diagnostic queries we are using the conditional probability P(cause|effect). Figure 9 shows a **diagnostic** query, calculating the probability that an attack (A=True) caused the alert to be triggered.

Predicting Potential Attack Given Evidence - Diagnostic			
Query	**Node Elimination Order**	**Evidence**	**Result**
A:T	IOD,M,H,FD,NB,AT,CL	AT:T	0.70563

Table 11: Predicting portential attack given evidence – diagnostic query.

When the alert is triggered, there is only a 70.56 percent chance of a true attack. Although the likelihood of a false positive alert trigger is very low (1%), because an attack not occurring is significantly more likely than an attack occurring, approximately 30% of alert triggers are the result of false positives. In isolation, this result appears dubious; however, when we consider the extremely low likelihood of an attack, this result makes perfect sense. Using the BN, the company can find the probability of causes for certain events that occur.

Ordering (Task 4)

To effectively examine the effect of ordering, I mindfully decided to check it on a query where no nodes could be pruned as this would allow us to see how much difference there is.on a whole BN level.

Query used to evaluate ordering	
Query	**Evidence**
A:T	CL:T AT:T

Table 12: The query used to evaluate ordering.

Figure 9 proves how every node is used in the ordering of this query. The ancestor (Attack) of evidence is pointed to with an orange pointer. The ancestors of the query node are pointed to with green pointers.

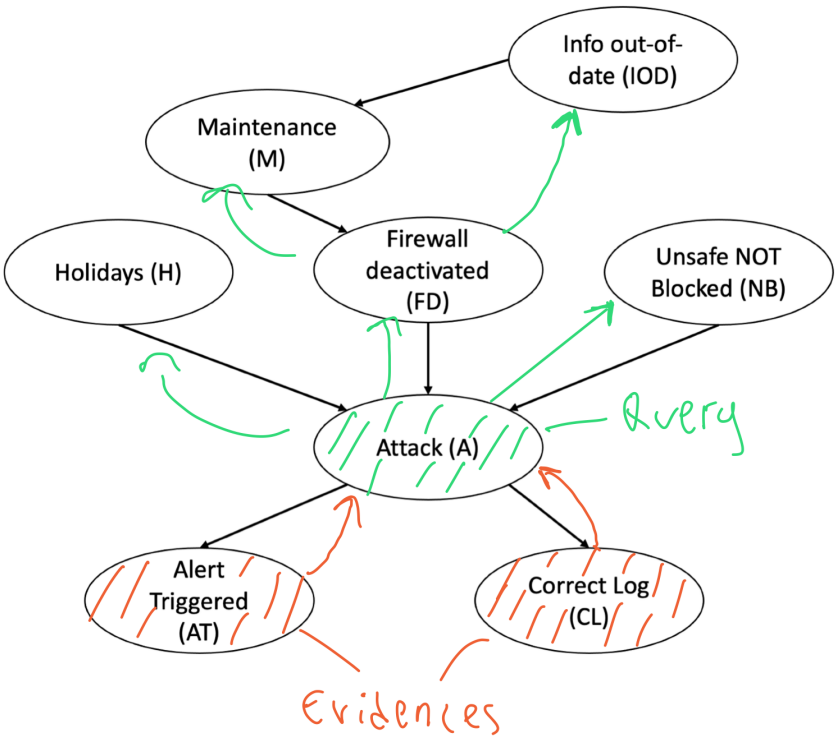


Figure 9: Pruning variables based on ancestors. No pruning done here.

Elimination Ordering Comparison - Average of 20 runs							
Query	**Node Elimination Order**	**Order Reasoning**	**Evidence**	**Running time (µs)**	**JoinMarginalise Operations**	**Truth Values Calculated**	**Output**
AT:T	AT, IOD, M, CL, FD, NB, H	Maximum Cardinality	CL:T AT:T	2074 µs	80	713	0.63850
AT:T	AT,M,CL,FD,IOD,NB,H	Random	CL:T AT:T	3908 µs	80	1265	0.63850

Table 10: Elimination ordering comparison between maximum cardinality algorithm and random order.

As previously explained, ordering plays a significantly role in the amount of computations. Here as evident the amount of truth values calculated and running time is significantly less using the maximum cardinality variable elimination than random. Furthermore, as expected, the number of joinMarginalise operations and output is the same between the two. We can conclude that ordering improves the alert system, enabling faster inferences by doing less calculations, while yielding the same outputs.

References:

[1] Russell, Stuart J. (Stuart Jonathan). Artificial Intelligence : a Modern Approach. Upper Saddle River, N.J. :Prentice Hall, 2010.