CS5011 A3

200010781

MARCH 30, 2022

[COMPANY NAME]
[Company address]

Contents

1 Introduction

This report describes the third assignment for CS5011 module. General information can be seen in the Table 1 below.

Table 1 General information of the assignment

Assignment Number	Assignment 23
Date of Submission	30 March 2022
Student ID	200010781
Word Count	1998

There are four in this assignment, including building four networks in part 1, making simple inferences in part 2, adding evidence in part 3, and deciding on an order in part 4 using max cardinality. The completion of each agent is described as Table 2 below.

Table 2 Completion of each part

Part 1	Build Network	attempted, fully working
Part 2	Simple Inferences	attempted, fully working
Part 3	Adding Evidence	attempted, fully working
Part 4	Deciding on an order	attempted, fully working

About running the source code, several steps are needed.

1. Enter the terminal.
2. Change to the CS5011_A3 directory.
3. Compile all source code

   javac *.java help/*.java agent/*.java
4. Run the source code

   java A3main <P1|P2|P3|P4> <NID>

   For example, if I want to do simple inferences on BNA network, I should use

   java A3main P2 BNA

   Then enter the required query and order.

2 Design and Implementation

2.1 Problem Overview

This assignment focuses on modelling and using Bayesian networks to reason with uncertainty. In this assignment, the PEAS model is described as

Table 3 below.

Table 3 PEAS model for assignment 2

Agent	Performance Measure	Environment	Actuator	Sensor
Uncertainty agent	Correct result, Number of join and marginalise	Bayesian Network	Making inferences	Using Bayesian rules

2.2 System Architecture

All source code can be found in the src directory. The main function is addressed in A3main.java file which locates in src. There are two packages in the src directory.

- **Help**

  Help package exists some auxiliary classes.

  - BayesianNetwork class

    There is an ArrayList of Node which stores all nodes in a network in the BayesianNetwork class. InducedGraph() method is used to create Induced graph in part 4.

  - CPT class

    There is a String[][] named TValue which aims at storing the true value in a CPT and a double[][] named PValue which aims at storing the probabilities in a CPT. TValue is one length bigger than PValue. This is because I store the labels in TValue[0], the first row.

  - Factor

    This class is used when doing join and marginalise. There exists an ArrayList of String named factors which stores the labels in the factor. For example, when dealing with f(A,B) in join and marginalise process, labels “A” and “B” will be added into the ArrayList<String> factors. Besides, there is a CPT in the Factor class, which represents the CPT of a factor.

  - Node

    I use a String named label to record the name of a node. There are three ArrayList<Node>, which represent the children, parents and neighbors of a node. There also exists a CPT, which is used to record the set CPT of a node from Part 1.

- **Agent**

  - P1

    This class mainly aims at creating four networks, including CNX, BNA, BNB, BNC.

- P2
  This class mainly aims at doing simple inferences based on networks built in P1.
- P3
  This class extends P2 with added ArrayList<String[]> evidence. It uses new varElimination() method and override createOrder() method.
- P4
  This class extends P3 with added new maxCardinality() method to decide the order of making inferences.

2.3 Implementation

2.3.1 Part 1

This part mainly focuses build four different networks, including CNX, BNA, BNB, and BNC. Since there are some probabilities that have not been given, I choose the probability that I think is reasonable for them. I build the CNX network as shown as Figure 1 below. According to the provided information, a planned maintenance can cause the firewall deactivated some time, which can increase the probability of having an attack in CNX system. If a maintenance is out of date, it can cause vulnerabilities in CNX system. Besides, holiday increases the intensity of attacks in a year, and there is some probability that some malicious websites being identified as safe. These can also increase the probability of having an attack in the system. Finally, there are some noises in the system, not all attacks will be triggered and not all system logs are correct information.

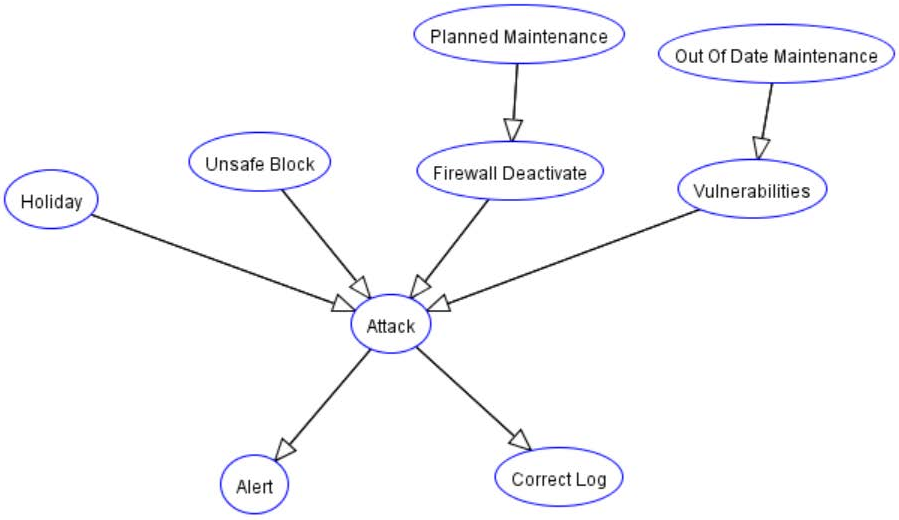

Figure 1 CNX Bayes Network

The selected probabilities of each event is illustrated as below.

- ✧ There are 45 days of holiday in 360 days. This means there is 12.5% probability that it is holiday in a year.
- ✧ There is 15% chance that some unsafe websites will be identified as safe.
- ✧ I assume that there is 20% probability that a maintenance is planned. Usually there is only 1% probability that the firewall is deactivated due to noises. If a maintenance is planned, there is 3% probability that the firewall will be deactivated.

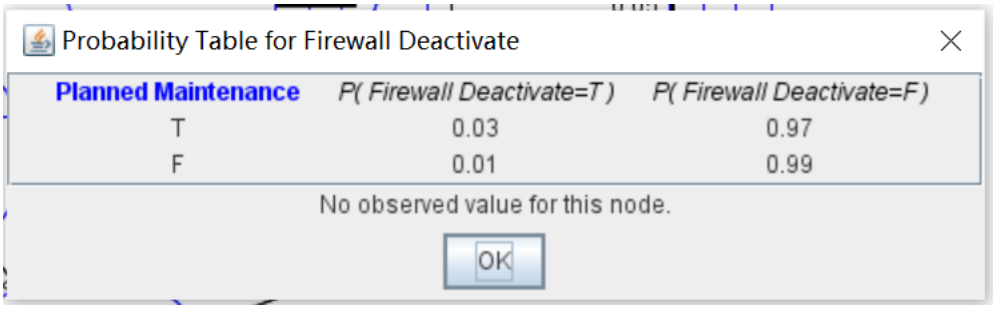

Figure 2 CPT of Firewall Deactivate

- ✧ I assume that there is 5% probability that a planned maintenance is out of date. Usually, there is only 1% probability that there are vulnerabilities in the system. According to the information provided in the specification, if a planned maintenance is out of date, the probability will increase to 5%.

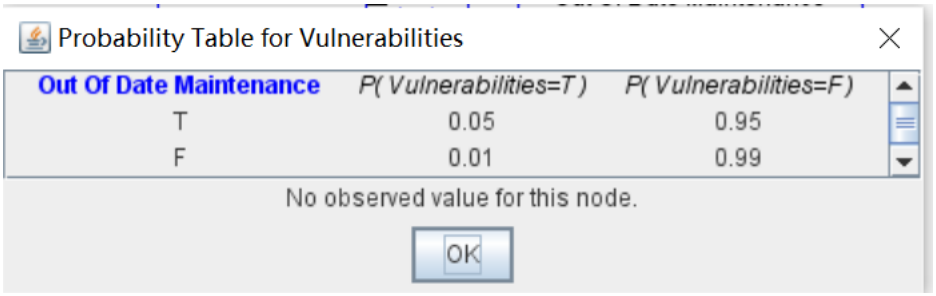

Figure 3 CPT of Vulnerabilities

- According to the requirements, if there is attack, there is 95% probability that the alert will be triggered. I assume that if there is no attack, the alert will be not triggered at all.

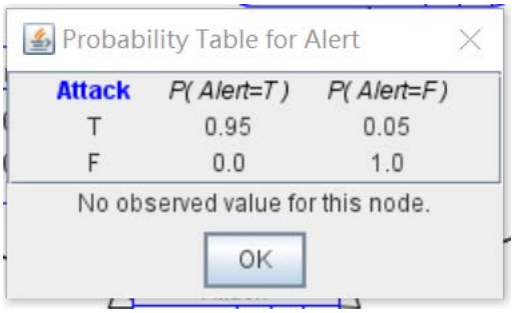

Figure 4 CPT of Alert

- According to the requirements, if there is attack, there is 70% probability that we get the correct log information. I assume that if there is no attack, we will not get any log information.

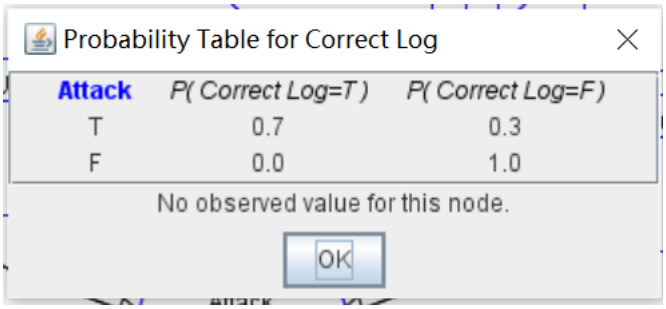

Figure 5 CPT of Correct Log

- The CPT of attack is a bit more complex than others. If the firewall is deactivated due to an out-of-date planned maintenance in holiday time, and accessing to unsafe websites cannot be blocked, there will be 30% that the CNX can be attacked. Otherwise, if in a totally opposite situation, there is only 1% that CNX can be attacked.

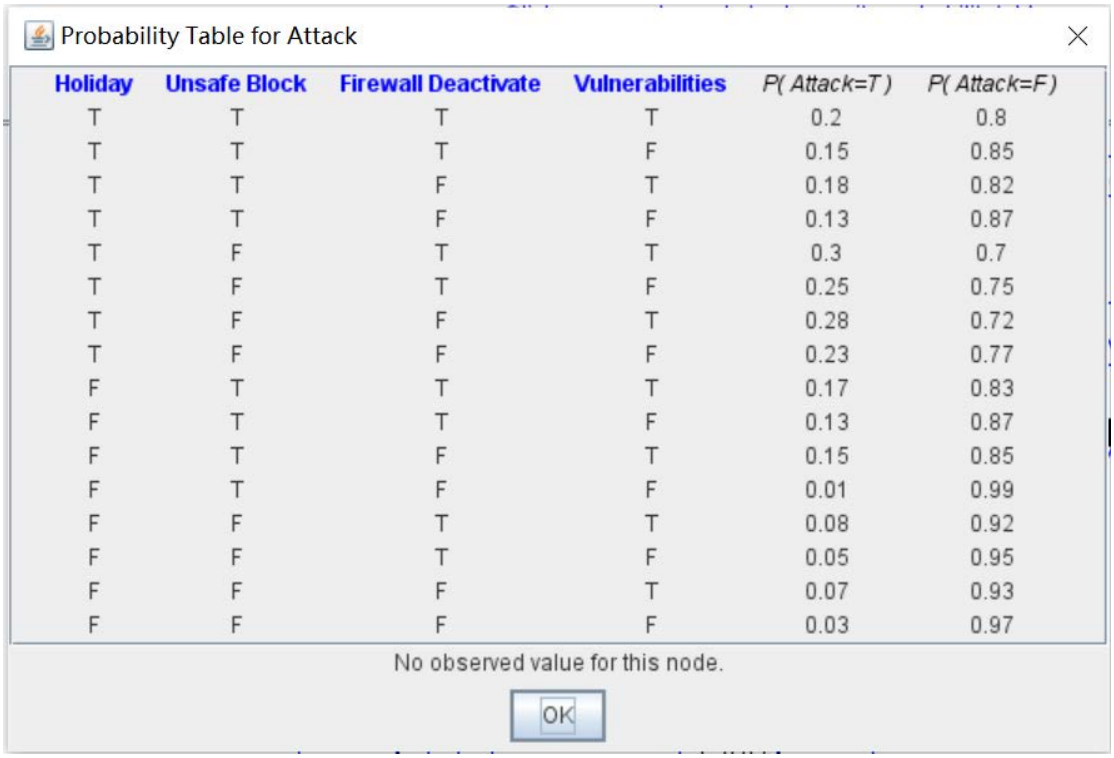

Figure 6 CPT of Attack

The overall network of CNX represented by bayes.jar is shown as below.

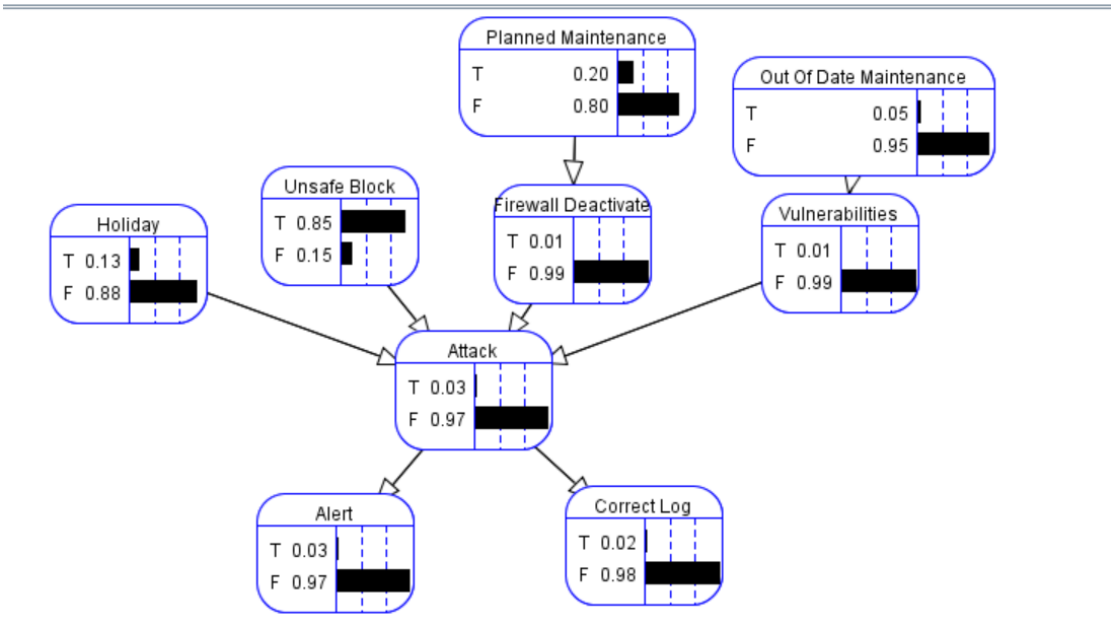

Figure 7 CNX Bayes Network Overview

2.3.2 Part 2

P2 class in agent package is used to implement part 2 requirements. When constructing the true value in CPT, I represent “T” as “0” and “F” as “1”. The reason is because the 0 and 1 can be regarded as the index of probability

value in binary format. For example, the true value of “011” is 3 in decimal. So the probability in double[][] PValue is double[3][0].

After constructing the desired network in P1, P2 class uses varElimination() method to do simple inferences. At first, I use createOrder() and createFactors() to create orders and factors. Then for each item in finalOrder list, if a Factor in factors list has the same label as the item, it will be added into toSumOut ArrayList<Factor>. Then, I call joinMarginalise() method to do join and marginalize processes. In joinMarginalise() method, I first estimate the number of factors in toSumOut ArrayList. If it is more than one, I call joinProcess() and margProcess() to do join and marginalise. But if it is only one, I only call margProcess() to marginalise the target item.

In joinProcess() method, I choose two factors fA and fB in the toSumOut list, and use findV1(), findV2() and findV3() to get V1, V2 and V3 respectively. Then get the union of these three, which is the new Factor variables. Finally, find desired probabilities in two factors and get the production of these probabilities as the new PValue for the new Factor. At the end, return the new Factor named as fC.

In margProcess() method, I use String[][] NewTValue and double[][] NewPValue to create the CPT for the new Factor. Int[] marker is used to mark whether a probability is already be marginalised. When getting the sum of two probabilities, I get the index of the target label I desire to marginalise at first, if the true value of it is “0”, I just get the index where the true value is “1”, so that I can use binary-decimal approach to get the two probabilities, and sum them together as the probability of new Factor CPT.

At last, find the result in the final Factor where the true value is the target value from user input and print it.

2.3.3 Part 3

P3 class in the agent package is used to address this part. I override the createOrder() method, adding nodes in the finalOrder that is evidence or ancestor of evidence nodes. varElimination() method is similar to part 2. The differences are that I project evidence for related factors using changePValue() method in the CPT class after creating orders and factors and I call Normalise() method in CPT class to do normalise.

At last, find the result in the final Factor where the true value is the target value from user input and print it.

2.3.4 Part 4

P4 class in the agent package is used to address this part. The main difference between P3 and P4 is that P4 decide the orders automatically. I use a method maxCardinality() to decide the orders in the inferences process. In this method, I use unmarked and marked lists to record which nodes are already been considered or not. I use for iterations to find the node in the unmarked with maximum number of marked neighbours and add it in the

order ArrayList. Finally, reverse the order and remove the query node to get the final order. Then return the finalOrder and use varElimination() in P3 to get the result.

3 Test Summary

In this assignment, I tested all test cases and select two cases in each network (BNA, BNB, BNC) using different methods.

- CNX network
  - ✧ **Part 2**

    In CNX network, there is about 7.3% probability that the CNX system is facing an attack. In Figure 8, it is clear that using different order will not affect the inference results.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 CNX
Query:
attack:T
Order:
holiday, planedMaintenance, outOfDateMaintenance, vulnerability, fireWallDeact, unsafeBlocked, alert, correctLog
0.07303
```

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 CNX
Query:
attack:T
Order:
unsafeBlocked, outOfDateMaintenance, fireWallDeact, vulnerability, correctLog, alert, holiday, planedMaintenance
0.07303
```

Figure 8 Test on CNX network in part 2

  - ✧ **Part 3**

    When doing a diagnostic query in CNX system as shown in the first picture of Figure 9, if we observe a correct log in the CNX system, there is about 57.8% probability that it is not in the holiday time. When doing a predictive query in the system as shown in the second picture of Figure 9, if we observe that it is not in the holiday time, there is about only 3.3% probability for getting the correct log information.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 CNX
Query:
holiday:F
Order:
unsafeBlocked, outOfDateMaintenance, fireWallDeact, vulnerability, correctLog, alert, attack, planedMaintenance
Evidence:
correctLog:T
0.57860
```

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 CNX
Query:
correctLog:T
Order:
unsafeBlocked, outOfDateMaintenance, fireWallDeact, vulnerability, attack, alert, holiday, planedMaintenance
Evidence:
holiday:F
0.03380
```

Figure 9 Test on CNX network in part 3

  - ✧ **Part 4**

    When doing the diagnostic query for holiday=F as in CNX part 3, the system automatically choose the order: “correctLog alert outOfDateMaintenance planedMaintenance unsafeBlocked fireWallDeact vulnerability attack”. The

result is the same as previous, about 57.8%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 CNX
Query:
holiday:F
Evidence:
correctLog:T
Final Order: correctLog alert outOfDateMaintenance planedMaintenance unsafeBlocked fireWallDeact vulnerability attack
0.57860
```

Figure 10 Test on CNX network in part 4

- BNA network
  - ✧ **Part 2**

In BNA network doing simple inferences for P(D=T) in P2, the results show that there are about 57% probabilities that variable D is True in both “ABC” and “BCA” orders with no evidence. This shows that the order will not affect the result.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 BNA
Query:
D:T
Order:
A,B,C
0.57050
```

Figure 11 Test case BNAP2q1 result

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 BNA
Query:
D:T
Order:
B,C,A
0.57050
```

Figure 12 Test case BNAP2q2 result

✧ **Part 3**

In BNA network doing inference for P(D=T|A=T) in P3, the result shows that if A is True, there is about 54.2% probability that variable D is True in “ABC” order.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 BNA
Query:
D:T
Order:
A,B,C
Evidence:
A:T
0.54200
```

Figure 13 Test case BNAP3q1 result

When doing inference for P(D=T|A=T, B=T) in P3, the result shows that if A and B are True, there is about 58% probability that variable D is True. Comparing with previous example, B is True can increase the probability that D is True.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 BNA
Query:
D:T
Order:
A,B,C
Evidence:
A:T B:T
0.58000
```

Figure 14 Test case BNAP3q2 result

✧ **Part 4**

In BNA network doing inference for P(D=T|A=T) in P4, I use max cardinality to determine the order. Comparing with previous example of BNAP3q1, the order changes to “CBA” automatically. However, the results remain the same, is about 54.2%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 BNA
Query:
D:T
Evidence:
A:T
Final Order: A B C
0.54200
```

Figure 15 Test case BNAP3q1 result using automatically determined order

Similarly, comparing with previous example of BNAP3q1, the order changes to “CBA” automatically when doing inference for P(D=T|A=T, B=T) in P4. But the result is the same, which is about 58%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 BNA
Query:
D:T
Evidence:
A:T B:T
Final Order: A B C
0.58000
```

Figure 16 Test case BNAP3q2 result using automatically determined order

- BNB network

✧ **Part 2**

In BNB network doing simple inferences for P(N=T) in P2, the result shows that there is about 40% probability that variable N is True in “JLKMO” order with no evidence.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 BNB
Query:
N:T
Order:
J,L,K,M,O
0.39864
```

Figure 17 Test case BNBP2q1 result

When doing simple inferences for P(M=T) in P2, the result shows that there is about 49.6% probability that variable M is True in “JLKMO” order with no

evidence.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 BNB
Query:
M:T
Order:
J, L, K, M, O
0.49660
```

Figure 18 Test case BNBP2q2 result

✧ **Part 3**

In BNB network doing inference for P(N=T|J=T) in P3, the result shows that if J is True, there is about 43.3% probability that variable N is True in "JLKMO" order.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 BNB
Query:
N:T
Order:
J, L, K, M, O
Evidence:
J:T
0.43360
```

Figure 19 Test case BNBP3q3 result

When doing inference for P(K=T|O=T) in P3, the result shows that if O is True, there is about 54.39% probability that variable K is True.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 BNB
Query:
K:T
Order:
O, J, M, L, N
Evidence:
O:T
0.54385
```

Figure 20 Test case BNBP3q1 result

✧ **Part 4**

In BNB network doing inference for P(N=T|J=T) in P4, the order use is "OMLKJ". Comparing with previous example of BNBP3q3, the result remains the same, about 43.3%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 BNB
Query:
N:T
Evidence:
J:T
Final Order: O J L K M
0.43360
```

Figure 21 Test case BNBP3q3 result using automatically determined order

In BNB network doing inference for P(K=T|O=T) in P4, the order use is "ONMLJ". Comparing with previous example of BNBP3q1, the result remains the same, about 54.39%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 BNB
Query:
K:T
Evidence:
O:T
Final Order: O N M L J
0.54385
```

Figure 22 Test case BNBP3q1 result using automatically determined order

- BNC network
  - ✧ **Part 2**

In BNC network doing simple inferences for P(U=T) in P2, the result shows that there is about 42.7% probability that variable U is True in “PRZSQV” order with no evidence.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 BNC
Query:
U:T
Order:
P, R, Z, S, Q, V
0.42755
```

Figure 23 Test case BNCP2q1 result

When doing simple inferences for P(S=T) in P2, the result shows that there is about 49.6% probability that variable S is True in “PURZQV” order with no evidence.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P2 BNC
Query:
S:T
Order:
P, U, R, Z, Q, V
0.49660
```

Figure 24 Test case BNCP2q2 result

  - ✧ **Part 3**

In BNC network doing inference for P(Z=T|R=T, U=T) in P3, the result shows that if R and U are True, there is about 43.3% probability that variable Z is True in “PURSQV” order.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 BNC
Query:
Z:T
Order:
P, U, R, S, Q, V
Evidence:
R:T U:T
0.43368
```

Figure 25 Test case BNCP3q1 result

When doing inference for P(P=T|Z=T) in P3, the result shows that if Z is True, there is about 5.5% probability that variable P is True.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P3 BNC
Query:
P:T
Order:
U, R, Z, S, Q, V
Evidence:
Z:T
0.05509
```

Figure 26 Test case BNCP3q2 result

✧ **Part 4**

In BNC network doing inference for P(Z=T|R=T, U=T) in P4, the automatically determined order is “USVRQP”. Comparing with previous example of BNCP3q1, the result remains the same, about 43.3%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 BNC
Query:
Z:T
Evidence:
R:T U:T
Final Order: U P R Q S V
0.43368
```

Figure 27 Test case BNCP3q1 result using automatically determined order

In BNC network doing inference for P(P=T|Z=T) in P4, the automatically determined order is “UZSVRQ”. Comparing with previous example of BNCP3q2, the result remains the same, about 5.5%.

```
C:\Users\DaisyDai\Desktop\JAVA-StAndrews\S2\CS5011_A3\src>java A3main P4 BNC
Query:
P:T
Evidence:
Z:T
Final Order: U Z S V R Q
0.05509
```

Figure 28 Test case BNCP3q2 result using automatically determined order

4 Evaluation and Conclusion

From previous section 3, it is clear that different orders will not affect the result of query. This part mainly discuss how order affects the inner computation of query process. I record the number of entries in join process to show how order can affect the performance of the system as shown in Table 4. The green numbers are the result from order formed by max Cardinality method.

In CNX network, the best order with the least number of entries is the order: “Alert,outOfDateMaintenance,planedMaintenance,unsafeBlocked,fireWallDeact,vulnerability,holiday,attack”. In this network, Node Attack has four parents so it’s factor is f(attack, holiday, unsafeBlocked, vulnerability, fireWallDeact). Therefore, it is better to deal with it at the end of the order.

In BNA network, the best order is “A,B,C”. At first, we join f(A) and f(B,A) and marginalise A to form f(B).Then we join f(B) and f(C,B) to get f(C). Finally we join and marginalise f(C) and f(C,D) to form f(D) to get the answer. However, if we use “C,B,A” order. At first, we join f(C,B) and f(C,D) to f(B,D), where the

number of entries is 8. Then we join f(B,A) and f(B,D) to f(A,D), where the number of entries is 8 as well. Finally we join f(A) and f(A,D) to get f(D), where the number of entries is 4. This increases the number of entries from 12 to 20. Similar to BNC and BNB network, it is better to put the node with more parents in the behind of the order list.

According to the table 4, it is clear that the order will not affect the result, but good order provide us better performance.

Table 4 Performance of different order in different networks

Network	Query	Evidence	Order	Num of Entries	Probability
CNX	correctLog:T	holiday:T	Alert,outOfDateMaintenance, planedMaintenance,unsafeBlocked, fireWallDeact,vulnerability,holiday,attack	72	0.17234
CNX	correctLog:T	holiday:T	attack,holiday,planedMaintenance, outOfDateMaintenance,vulnerability, fireWallDeact,unsafeBlocked,alert	132	0.17234
CNX	correctLog:T	holiday:T	vulnerability,fireWallDeact,unsafeBlocked, alert,attack,holiday,planedMaintenance, outOfDateMaintenance	230	0.17234
BNA	D:T	A:T	A,B,C	12	0.54200
BNA	D:T	A:T	B,A,C	16	0.54200
BNA	D:T	A:T	C,B,A	20	0.54200
BNB	N:T	K:T	O,J,L,K,M	20	0.45200
BNB	N:T	K:T	K,J,L,M,O	32	0.45200
BNB	N:T	K:T	K,M,L,J,O	44	0.45200
BNC	P:T	Z:T	U,Z,S,V,R,Q	34	0.05509
BNC	P:T	Z:T	S,V,R,Q,U,Z	66	0.05509
BNC	P:T	Z:T	S,R,Q,U,Z,V	74	0.05509