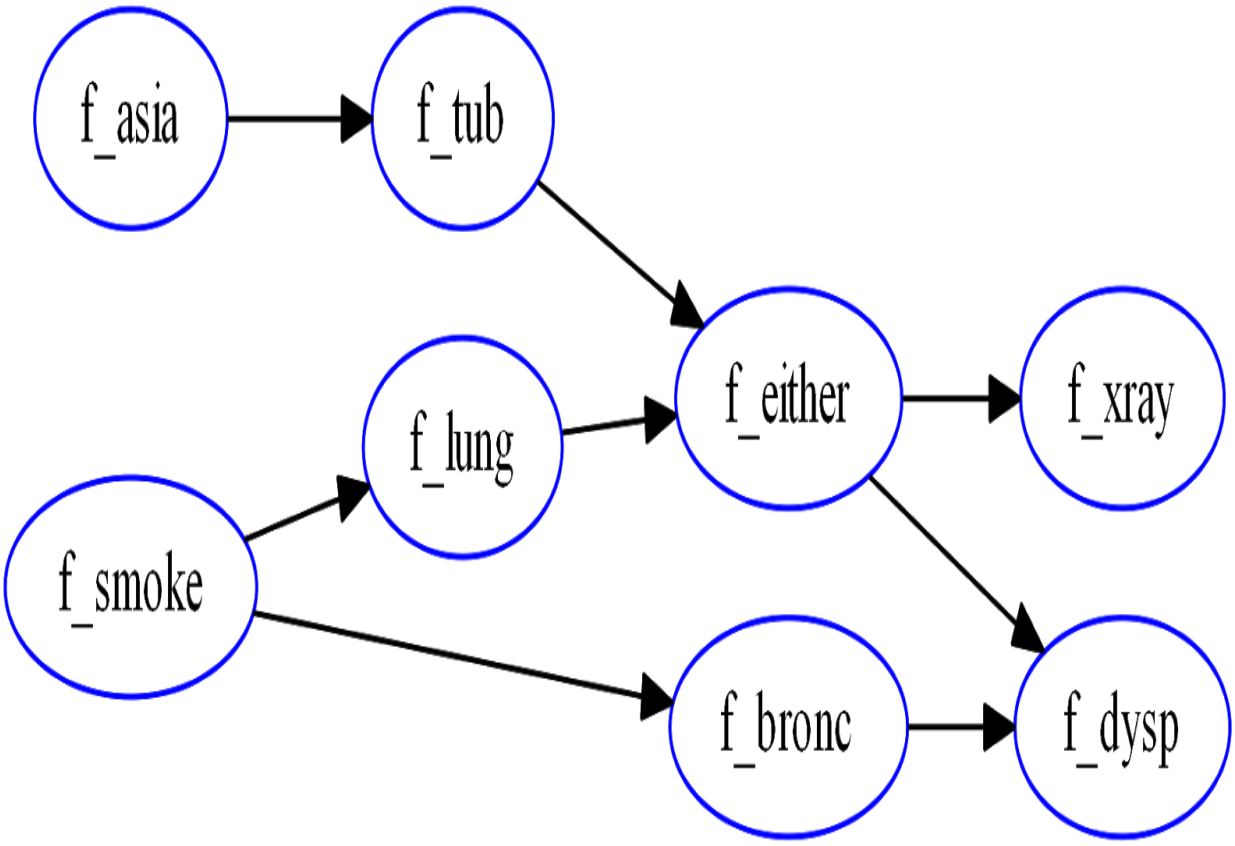
PROBLEM SET 2 REPORT

Introduction to Pattern Recognition- CSE-555



Jayant Solanki

UBIT Name: jayantso
Person Number: 50246821
SPRING 2018 CSE

INTRODUCTION

This report contains the implementation details of Problem Set 2 i.e., making exact inferences about probabilistic graphical models using the state-of-the-art graphical model packages in R and Python programming languages, and understanding of those exact algorithms. We worked on the Chest Clinic Graphical Model of Lauritzen and Spiegelhalter (1988) (illustrated in Figure 1 below).

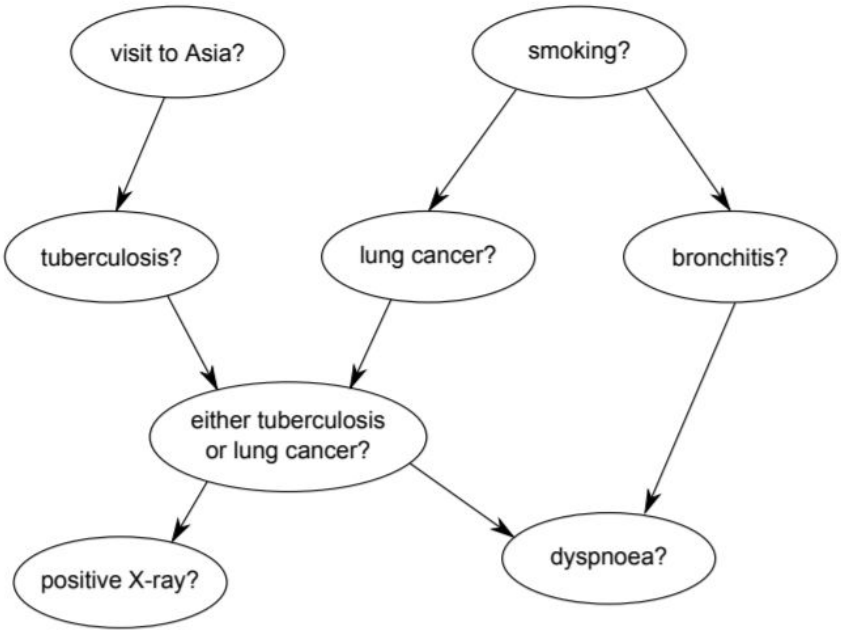


Figure 1

OBJECTIVES ACHIEVED

We used the Chest Clinic Graphical Model and then moralize it to create the Moral Graph, Triangulated it to create the Triangulated Graph, created Junction Tree based on that Triangulated Graph. We verified that **running intersection property** is satisfied in the Junction Tree, and then used the Message-Passing algorithm to find inference.

IMPLEMENTATION

BACKGROUND

(Asia) Chest Clinic[1]: Lauritzen and Spiegelhalter (1988) motivate the chest clinic example as follows:

> "Shortness–of–breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X–ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea."

Bayes Nets:

A Bayes net is a model. It reflects the states of some part of a world that is being modeled and it describes how those states are related by probabilities. The model might be of our house, or our car, our body, our community, an ecosystem, a stock-market, etc. Absolutely anything can be modeled by a Bayes net. All the possible states of the model represent all the possible worlds that can exist, that is, all the possible ways that the parts or states can be configured. The car engine can be running normally or giving trouble. It's tires can be inflated or flat. Our body can be sick or healthy, and so on.

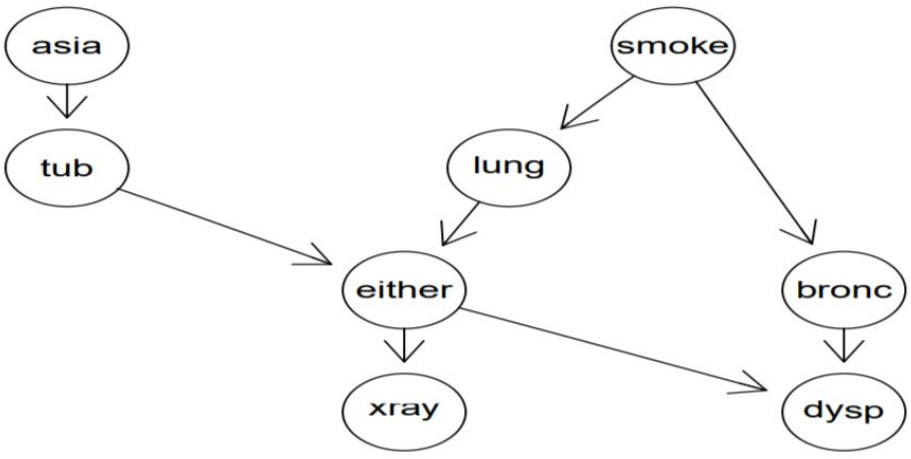


Figure 2

[1] https://www.norsys.com/tutorials/netica/secA/tut_A1.htm

Above in the figure 2, is an example of simple Bayes net called **Asia.** It is a simplified version of a network that could be used to diagnose patients arriving at a clinic. Each node in the network corresponds to some condition of the patient, for example, "Visit to Asia" indicates whether the patient recently visited Asia. The arrows (also called links) between any two nodes indicate that there are probability relationships that are known to exist between the states of those two nodes. Thus, smoking increases the chances of getting lung cancer and of getting bronchitis. Both lung cancer and bronchitis increase the chances of getting dyspnea (shortness of breath). Both lung cancer and tuberculosis, but not usually bronchitis, can cause an abnormal lung x-ray. And so on.

Bayes Rule:

For any two events, A and B,

$$p(B|A) = p(A|B)\ x\ p(B) / p(A)$$

where ***'p(A)'*** is "the probability of ***A***", and ***'p(A|B)'*** is "the probability of ***A*** given that ***B*** has occurred".

Building the Network:

A Bayesian network is a special case of graphical independence networks. In this section we outline how to build a Bayesian network. The starting point is a probability distribution factorising according to a DAG with nodes V . Each node $v \in V$ has a set $pa(v)$ of parents and each node $v \in V$ has a finite set of states. A joint distribution over the variables V can be given as

$$p(V) = Y\, v \in V\, p(v|pa(v)) \quad (1)$$

where $p(v|pa(v))$ is a function defined on $(v, pa(v))$. This function satisfies that $P\, v * p(v = v * |pa(v)) = 1$, i.e. that for each configuration of the parents $pa(v)$, the sum over the levels of v equals one. Hence $p(v|pa(v))$ becomes the conditional distribution of v given $pa(v)$. In practice $p(v|pa(v))$ is specified as a table called a conditional probability table or a CPT for short. Thus, a Bayesian network can be regarded as a complex stochastic model built up by putting together simple components (conditional probability distributions).

Thus the DAG in **Figure 2** dictates a factorization of the joint probability function as:

$$p(V) = p(\alpha)p(\sigma)p(\tau\ |\ \alpha)p(\lambda\ |\ \sigma)p(\beta\ |\ \sigma)p(|\ \tau,\ \lambda)p(\delta\ |,\ \beta)p(\xi\ |) \quad (2)$$

In **(2)** we have α = asia, σ = smoker, τ = tuberculosis, λ = lung cancer, β = bronchitis, ε = either tuberculosis or lung cancer, δ = dyspnoea and ξ = xray. Note that ε is a logical variable which is true if either τ or λ are true and false otherwise.

Moral Graph:

A moral graph is used to find the equivalent undirected form of a directed acyclic graph (dag). It is a key step of the junction tree algorithm, used in belief propagation on graphical models[2].

Triangulated Graph:

Break every cycle spanning 4 or more nodes into sub-cycles of exactly 3 nodes by adding arcs to the moral graph.

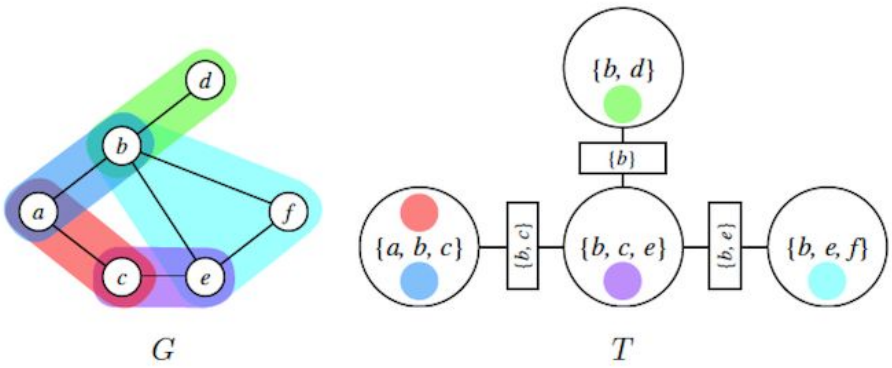
Finding Cliques:

Identifies the (clusters) cliques C_1.....C_k of the triangulated graph, i.e., maximal subsets of nodes in which each element is adjacent to all the others.

Junction Tree:

A tree in which each clique is a node, and adjacent cliques/clusters are linked by arcs and has these three properties:
1. **singly connected**: there is exactly one path between each pair of clusters.
2. **covering**: for each clique A of G there is some cluster C such that A⊆C.
3. **running intersection**: for each pair of clusters B and C that contain i, each cluster on the unique path between B and C also contains i.

Below G is the graph and T is the junction tree of the G.



[2] https://en.wikipedia.org/wiki/Moral_graph

Figure 2.1

Part 1 of the Problem Set has been implemented in both R and Python 2.7, Part 2 has been done in Python 2.7 only. Codes are available in the ipython notebooks inside the **code** folder.

1. **Creating the Network:**
    a. In Jupyter Notebook, the code for creating network in R is provided in the Problem Set itself.

```
yn <- c("yes","no")
a <- cptable(~asia, values=c(1,99), levels=yn)
t.a <- cptable(~tub | asia, values=c(5,95,1,99), levels=yn)
s <- cptable(~smoke, values=c(5,5), levels=yn)
l.s <- cptable(~lung | smoke, values=c(1,9,1,99), levels=yn)
b.s <- cptable(~bronc | smoke, values=c(6,4,3,7), levels=yn)
e.lt <- cptable(~either | lung:tub,values=c(1,0,1,0,1,0,0,1),levels=yn)
x.e <- cptable(~xray | either, values=c(98,2,5,95), levels=yn)
d.be<-cptable(~dysp|bronc:either, values=c(9,1,7,3,8,2,1,9),levels=yn)
cpt.list <- compileCPT(list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be))
```

    b. For Python, bif_parser and bayesian library from **eBay** has been used to create the network.

```
name = 'asia' #it is the name of the bif file
module_name = bif_parser.parse(name)
module = __import__(module_name)
bg = module.create_bbn()#creating the network
```

2. **Drawing of Moral graph, Triangulated Graph and Junction Tree:**
    a. Using the reference provided in the course slides I created graphs in R.

```
bnet = grain(cpt.list)
bnet = compile(bnet)
plot(bnet$dag)#original dag
plot(bnet$dag)#original dag
plot(moralize(bnet$dag))#moralize form
plot(triangulate(moralize(bnet$dag)))#triangulate form
plot(jTree(triangulate(moralize(bnet$dag))))# junction tree
```

    b. Using the reference provided in the course slides I created graphs in Python.

```
#moral
gu=make_undirected_copy(bg)
m1=make_moralized_copy(gu,bg)
s2=m1.get_graphviz_source()
Image(filename=show_graphgiz_image(s2))#showing moral graph

#triangulation
cliques, elimination_ordering = triangulate(m1, priority_func)
s2=m1.get_graphviz_source()
Image(filename=show_graphgiz_image(s2))#showing triangulated graph

#junction tree
jt=bg.build_join_tree()
sf=jt.get_graphviz_source()
Image(filename=show_graphgiz_image(sf))#showing the junction tree
```

3. **Running Intersection Property**

   a. The tree satisfies the running intersection property: if a node belongs to two cliques C_i and C_j , it must be also included in all the cliques in the (unique) path that connects C_i and C_j. Explanation is shown in the **result** section below

4. **Calculating joint probability:**
   a. We set the evidences for **asia** and **xray** to **yes** and found the joint probability for lungs, tub and bronc where all these three are yes, using the following codes below:

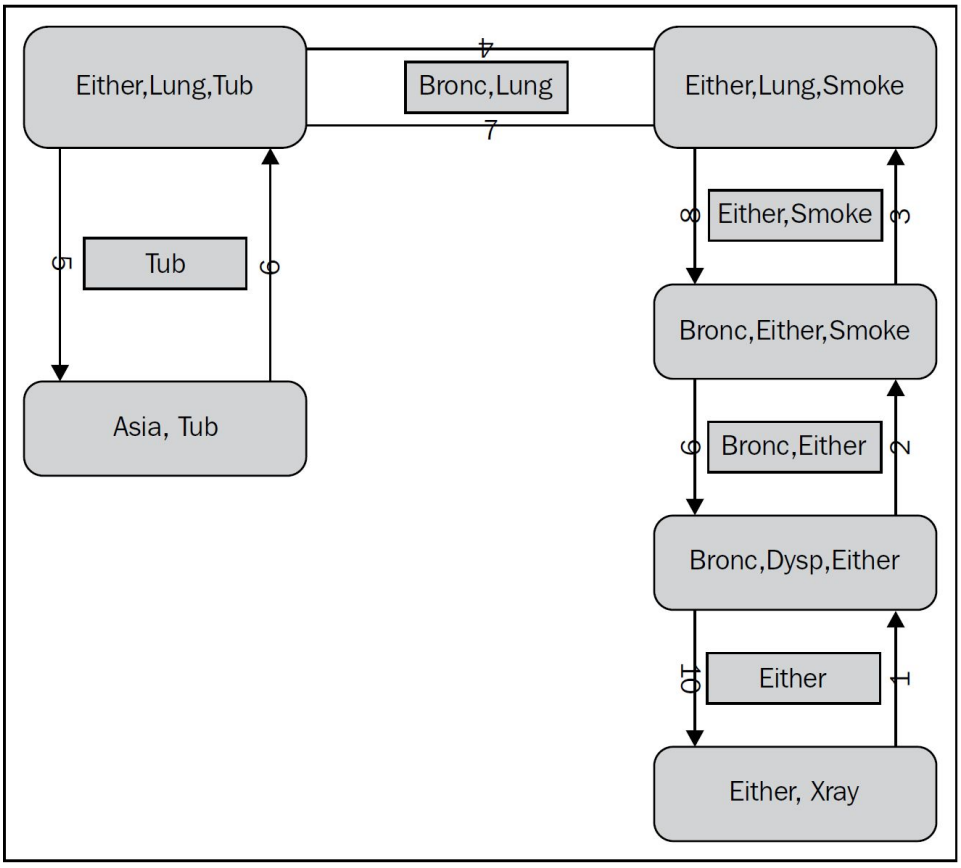
```
bnet.ev = setEvidence(bnet, nodes='asia',states="yes")

#creating new network by setting evidences asia and xray to yes

bnet.ev = setEvidence(bnet.ev, nodes='xray',states="yes")

querygrain(bnet.ev,nodes=c('tub','lung','bronc'))

cat("Displaying the joint probability for tub =yes, lung = yes and bronc = yes: ", jp[1])
```

5. **Distribution of variables in the Junction Tree:**
   a. Description for the distribution of right hand side of "p(V) = p(a)p(t |

a)p(s)p(l | s)p(b | s)p(e | t, l)p(d | e, b)p(x | e)" is shown in the result section below

6. **Message passing using the terms in the Chest clinic[3]:**
    a. We consider conditional probability tables (CPTs) as potentials/interactions that is the **q-functions.**
    b. p(V) = p(a)p(t | a)p(s)p(l | s)p(b | s)p(e | t, l)p(d | e, b)p(x | e)
    c. p(V) = q(a)q(t, a)q(s)q(l, s)q(b, s)q(e, t, l)q(d, e, b)q(x, e)
    d. Simplifying the q-functions, setting q(t, a) ← q(t, a)q(a) and q(l, s) ← q(l, s)q(s)
    e. p(V) = q(t, a)q(l, s)q(b, s)q(e, t, l)q(d, e, b)q(x, e)
    f. We have p(V) factoring according to this chordal graph as p(V) = q(t, a)q(l, s)q(b, s)q(e, t, l)q(d, e, b)q(x, e) where where q(l, s; b) = q(l, s)q(b, s) and q(l, b, e) 1.
    g. After making the cliques required to run message passing, we proceed to the next stage. In this stage, there are two messages sent between each pair of cliques: one in a forward and another in a reverse pass.
    h. The message sequence is shown in the below image for each clique, each showing forward and reverse pass:

[3] http://people.math.aau.dk/~sorenh/misc/2014-useR-GMBN/bayesnet-slides.pdf



Figure[4] 3

i. Messages are run using jt.propogate() method in python. Message passing between all the cliques will stop when the beliefs have converged.
j. Once the message passing is done, we are left with a junction tree which has consistent beliefs (messages) in all its cliques.
k. If query a particular variable in all the clusters in which it is present, using the python code, we get the same marginal values for that variable. Values have been shown in the Result section below. This indicates that the beliefs for all the variables are consistent across all clusters and sepsets shown in the **Figure** 7. Hence message passing algo indeed gives clusters marginals.

7. **Finding joint probability with Markov Chain Monte Carlo[5]:**

[4] https://ublearns.buffalo.edu/bbcswebdav/pid-4499403-dt-content-rid-18203418_1/xid-18203418_1
[5] Markov Chain Monte Carlo Lecture Notes, Charles J. Geyer

a. A Markov chain is a discrete time stochastic process $X_1, X_2, \ldots$ taking values in an arbitrary state space and having the property that the conditional distribution of X_{n+1} given the past, $X_1, \ldots, X_n$, depends only on the present state X_n.
b. The specification of a Markov chain model has two pieces, the initial distribution and the transition probabilities. The initial distribution is the marginal distribution of X_1 similar to what we calculated for PArt 2 of the Problem set 2. The transition probabilities specify the conditional distribution of X_{n+1} given X_n. Since we always assume stationary transition probabilities, this is just one conditional distribution, the same for all n.
c. By mathematical induction, above two pieces determine the marginal distribution of $X_1, \ldots, X_n$ for any n. The base of the induction is obvious, the marginal distribution of X_1 is the initial distribution. Assuming the distribution of $X_1, \ldots, X_{n-1}$ is known, the distribution of $X_1, \ldots, X_n$ is determined by the usual:

   Joint probability = conditional × marginal

   formula when densities exist, where "marginal" refers to the distribution of $X_1, \ldots, X_{n-1}$, "joint" refers to the distribution of $X_1, \ldots, X_n$, and "conditional" refers to the distribution of X_n given $X_1, \ldots, X_{n-1}$, which by the Markov property depends on X_{n-1} alone and is the specified transition probability.

DIRECTORY LAYOUT

There are 3 folders under **Assignment-2** directory:

- **code :**
  1. **bayesian folder:** contains the python library function for building bayesian model using bif_parser
  2. **ProblemSet2-solution-in-Python27.ipnb**: Run this ipython file for seeing the solution for part 1 and part 2
  3. **Part-1-solution-in-R.ipynb**: run this ipython file for seeing the part 1 output in R code
  4. **asia.bif**: contains the model description for chest clinic graph

5. **bif_parser.py**[6]: for parsing the asia.bif file

- **Outputs:** Contains moral graph, junction tree, triangulated graph and original model pictures.
- **Report:** Contains the word doc file of the documentation for Problem Set 2.

INSTALLATION

1. **For the R code in Jupyter Notebook for Windows 10**

   In the jupyter notebook type:

   source("http://bioconductor.org/biocLite.R")
   biocLite(c("graph", "RBGL", "Rgraphviz"))
   install.packages("gRain",dependencies=TRUE, repos ="http://cran.us.r-project.org")
   install.packages("Rgraphviz")

2. **For the Python code in Jupyter Notebook for Windows 10**

   In the anaconda command prompt type:

   conda install -c anaconda graphviz
   #add the path variable to environment file,
   #C:\Users\jayan\Miniconda3\pkgs\graphviz-2.38.0-4\Library\bin\
   #C:\Users\jayan\Miniconda3\pkgs\graphviz-2.38.0-4\Library\bin\graphviz\
   #path may vary depending upon where the graphviz is installed
   Close and reopen the anaconda command prompt (as python kernel restart is required) and type:
   pip install pydot
   pip install prettytable
   All set now and open the Jupyter Notebook

RESULTS

PART 1

Original Dag

[6] https://github.com/eBay/bayesian-belief-networks

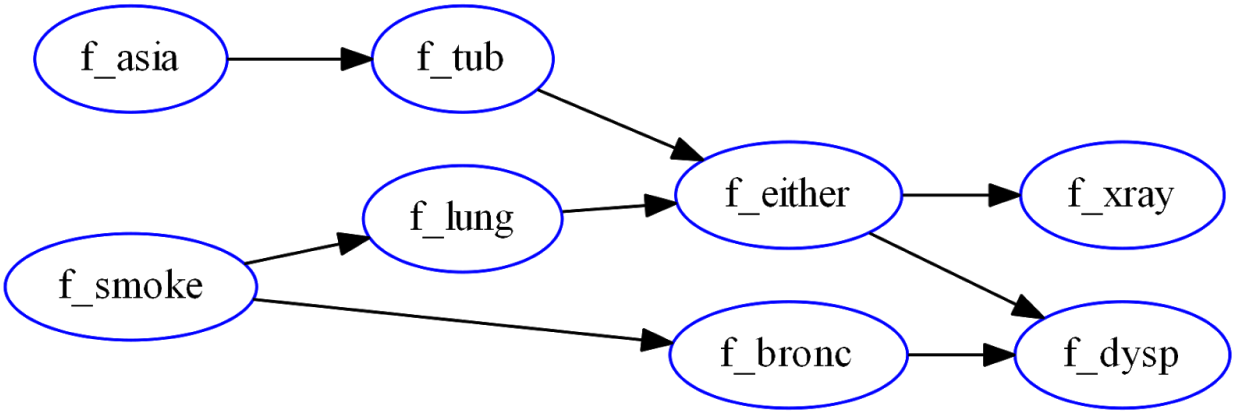


Figure 4

Moral Graph

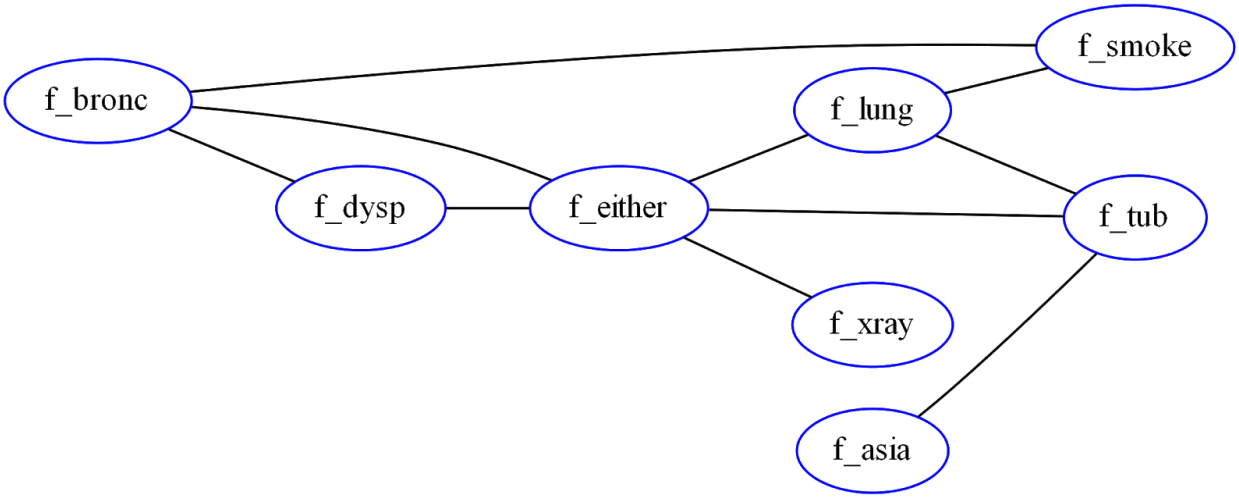


Figure 5

Triangulated Graph

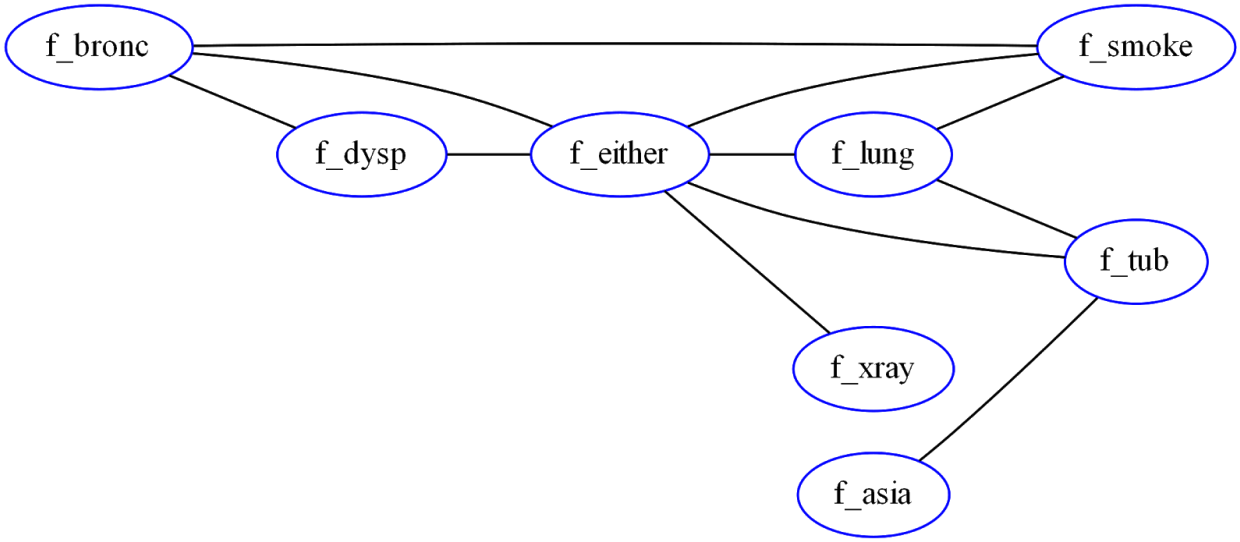


Figure 6

Junction Tree

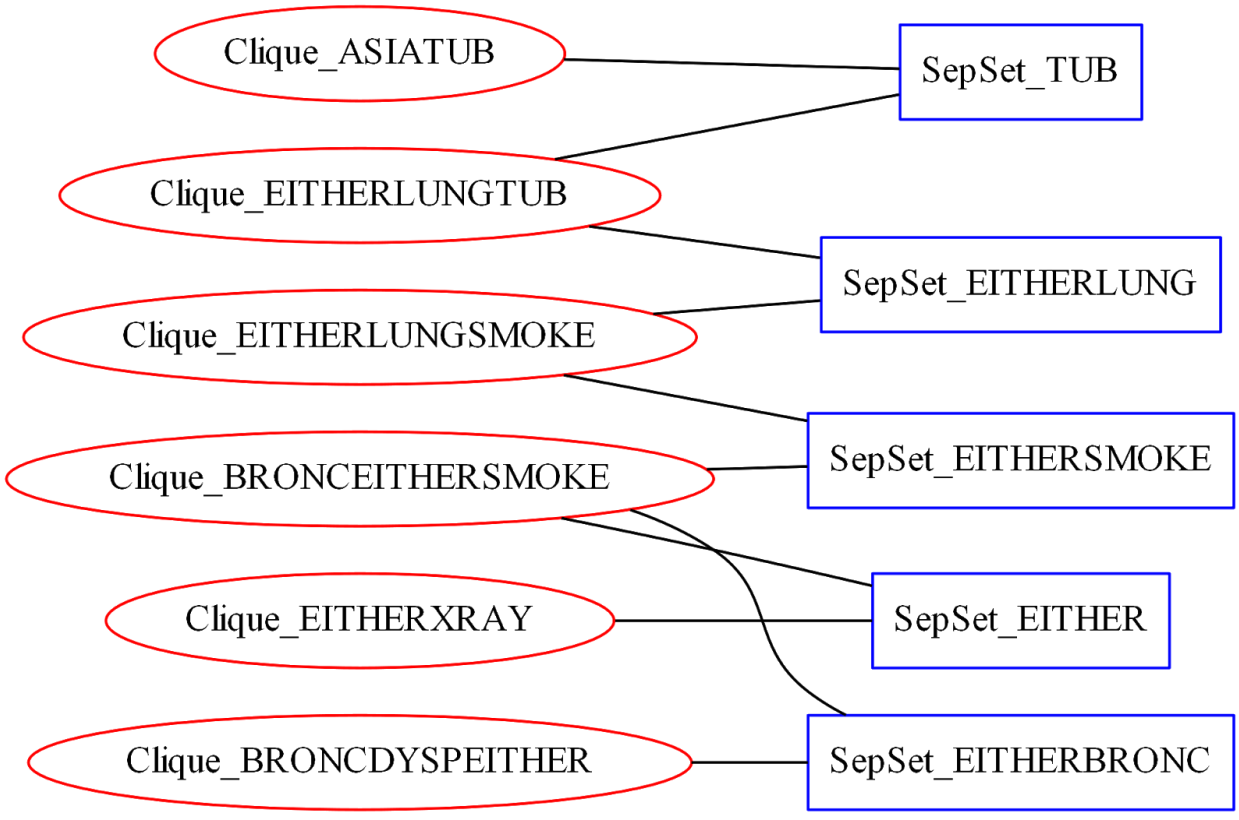


Figure 7

We can see from the variable names that the sepsets contain the intersection points of variables from the cliques that they connect with. For example, the variables in the cliques **{Bronc, Either, Smoke}** and **{Bronc,Dysp, Either}** are the variables in the **{Bronc, Either}** sepset.

In above Junction tree we can see that the Running Intersection Property is validated. For example, for the cliques/cluster pair **{Bronc, Either, Smoke}** and **{Lung, Tub, Either}**, which has the common variable of **Either**, there is a cluster **{Either, Lung, Smoke}** in the path connecting the above pair, that cluster is also having the same variable **Either** in it. Similarly clusters **{Either, Lung, Smoke} and {Either, Xray}** have a cluster **{Bronc, Either, Smoke}** in the their path which has variable **Either** common in all, same pattern in seen in cluster pairs having common variables , hence satisfying the **Running Intersection Property** in above Junction Tree.

The joint probability for tub =yes, lung = yes and bronc = yes, given evidence asis = yes and xray = yes is: **0.01063804**

PART 2

Description of the distribution of the the right hand side of "p(V) = p(a)p(t | a)p(s)p(l | s)p(b | s)p(e | t, l)p(d | e, b)p(x | e)" is shown in **Figure** 7.

Clique 1 = {a, t}
Clique 2 = {e, l, t}
Clique 3 = {e, l, s}
Clique 4 = {b,e,s}
Clique 5 = {e, x}
Clique 6 = {b, d, e}

Cluster marginals using the message passing algorithm on the Junction Tree is shown below:

bronc: yes 0.45 no 0.55
asia: yes 0.01 no 0.99
smoke: yes 0.5 no 0.5
dysp: yes 0.4359706 no 0.5640294
lung: yes 0.055 no 0.945
either: yes 0.064828 no 0.935172
xray: yes 0.11029004 no 0.88970996
tub: yes 0.0104 no 0.9896

SOFTWARE/HARDWARE USED

- Anaconda
- Jupyter Notebook
- Python 2.7 Environment based on Anaconda
- R 3.4 Environment based on Anaconda,
- Windows 10 System, Intel core i7 processor

- Python libraries: graphviz, pydot, prettytable, bayesian from eBay
- R libraries: gRain, Rgraphviz

REFERENCES

1. Ublearns
2. Stackoverflow.com
3. https://github.com/eBay/bayesian-belief-networks