

Project 7

Santiago Castro Dau, June Monge, Rachita Kumar, Sarah Lötscher

2022-04-28

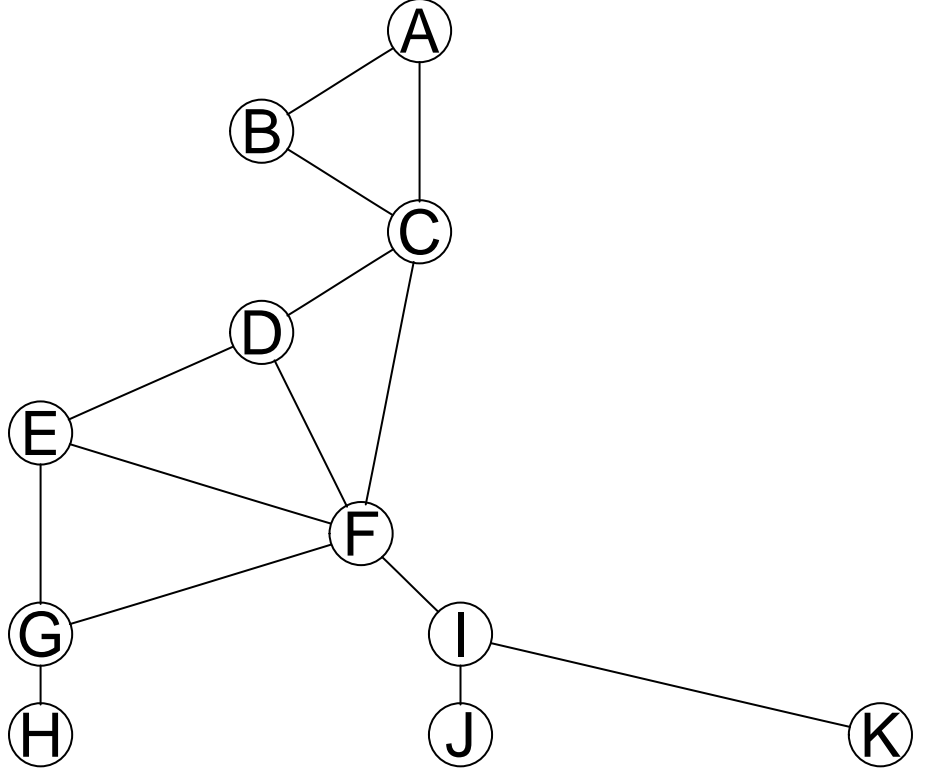
Problem 17: Junction Tree Algorithm

(a) Build the Junction Tree of the network.

```
adj_mat = matrix(1,
                  ncol = 11,
                  nrow = 11,
                  dimnames = list(LETTERS[seq( from = 1, to = 11 )],
                                  LETTERS[seq( from = 1, to = 11 )]))
adj_mat[lower.tri(adj_mat, diag = TRUE)] = 0

## Changing A
adj_mat[1, ] = c(0,1,1,0,0,0,0,0,0,0,0)
## Changing B
adj_mat[2, ] = c(1,0,1,0,0,0,0,0,0,0,0)
## Changing C
adj_mat[3, ] = c(1,1,0,1,0,1,0,0,0,0,0)
## Changing D
adj_mat[4, ] = c(0,0,1,0,1,1,0,0,0,0,0)
## Changing E
adj_mat[5, ] = c(0,0,0,1,0,1,1,0,0,0,0)
## Changing F
adj_mat[6, ] = c(0,0,1,1,1,0,1,0,1,0,0)
## Changing G
adj_mat[7, ] = c(0,0,0,0,1,1,0,1,0,0,0)
## Changing H
adj_mat[8, ] = c(0,0,0,0,0,0,1,0,0,0,0)
## Changing I
adj_mat[9, ] = c(0,0,0,0,0,1,0,0,0,1,1)
## Changing J
adj_mat[10, ] = c(0,0,0,0,0,0,0,0,1,0,0)
## Changing K
adj_mat[11, ] = c(0,0,0,0,0,0,0,0,1,0,0)

g = graphAM(adjMat = adj_mat, edgemode = 'undirected')
plot(g)
```



Moralize graph and triangulate

Build clique graph

Break cycles

(b) Write the joint probability $P(U)$ in terms of the cluster and separator potentials.

$$\begin{aligned}
 P(A, B, C, D, E, F, G, H, I, J) &= \sum_{A, B, C, D, E, F, G, H, I, J} P(U) \\
 &= \sum_{A, B, C, D, E, F, G, H, I, J} \phi(U) = \frac{\phi(ABC)\phi(DCF)\phi(DEF)\phi(EFG)\phi(GH)\phi(FI)\phi(IS)\phi(IK)}{\phi(C)\phi(DF)\phi(EF)\phi(G)\phi(F)\phi(I)^2}
 \end{aligned}$$

Problem 18: Benefit of storing messages

(a)

The formula for recursively computing the forward and backward messages is:

$$\begin{aligned}
 \mu_{\alpha}(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_{\alpha}(x_{n-1}) \\
 \mu_{\beta}(x_n) &= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \mu_{\beta}(x_{n+1})
 \end{aligned}$$

Initialization:

$$\mu_{\alpha}(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$$

$$\mu_{\beta}(x_4) = \sum_{x_5} \psi_{4,5}(x_4, x_5)$$

(b)

The computational complexity for computing the marginal probability $P(X_4 = 1)$ is $O(NK^2)$. For each node that can assume K values, since $(n-1)K^2$ computations have to be done for the forward message passing and $(N-n)K^2$ computations have to be done for the backward message passing, this sums to $(N-1)K^2$ computations which is of the order $O(NK^2)$

(c)

If the messages are stored, by doing one pass in the forward and one pass in the backward, the total number of steps is reduced to $2(N-1)K^2$, and therefore the order is $O(NK^2)$. In the general case, the number of steps for N nodes is $N(N-1)K^2$ and therefore the order of computational complexity is $O(N^2K^2)$

Problem 19 (data analysis): Message passing on a chain

(a) Store clique potentials in an R object

```
clique_potentials = array(dim = c(2, 2, 4), dimnames = list(c("0", "1"), c("0", "1"),
c("Psi12", "Psi23", "Psi34", "Psi45")))
```

compute clique potentials

```
X1 = c(1/3, 2/3)
X2 = c(4/5, 2/3)
X3 = c(5/7, 1/3)
X4 = c(3/5, 2/5)
X5 = c(1/2, 7/9)
```

```
clique_potentials = array(dim = c(2, 2, 4),
                          dimnames = list(c("0", "1"), c("0", "1"),
c("Psi12", "Psi23", "Psi34", "Psi45")))
```

```
clique_potentials[1,1,"Psi12"] = (2/3)*(1/5)
clique_potentials[1,2,"Psi12"] = (2/3)*(4/5)
clique_potentials[2,1,"Psi12"] = (1/3)*(1/3)
clique_potentials[2,2,"Psi12"] = (1/3)*(2/3)
clique_potentials[,,"Psi23"] = c(1-X3,X3)
clique_potentials[,,"Psi34"] = c(1-X4,X4)
clique_potentials[,,"Psi45"] = c(1-X5,X5)
```

```
clique_potentials[,,"Psi12"]
```

```
##           0           1
## 0 0.1333333 0.5333333
## 1 0.1111111 0.2222222
```

```
clique_potentials[,,"Psi23"]
```

```
##          0          1
## 0 0.2857143 0.7142857
## 1 0.6666667 0.3333333
clique_potentials[,,"Psi34"]
```

```
##      0      1
## 0 0.4 0.6
## 1 0.6 0.4
clique_potentials[,,"Psi45"]
```

```
##          0          1
## 0 0.5000000 0.5000000
## 1 0.2222222 0.7777778
```

(b) Computing forward messages

```
forward_messages = array(dim = c(5,2), dimnames = list(c("X1","X2", "X3","X4","X5"),c("0", "1")))
forward_messages["X1",] = c(1,1)
for(i in 2:5){
  forward_messages[i,]=forward_messages[i-1,]%*%clique_potentials[,i-1]
}
forward_messages
```

```
##          0          1
## X1 1.0000000 1.0000000
## X2 0.2444444 0.7555556
## X3 0.5735450 0.4264550
## X4 0.4852910 0.5147090
## X5 0.3570253 0.6429747
```

(d) Computing backward messages

```
backward_messages = array(dim = c(2,5),
                          dimnames = list(c("0", "1"),
                          c("X1","X2","X3","X4","X5")))
backward_messages[,,"X5"] = c(1,1)
for(i in 4:1){
  backward_messages[,i]=clique_potentials[,i]%*%backward_messages[,i+1]
}
backward_messages
```

```
##          X1 X2 X3 X4 X5
## 0 0.6666667 1 1 1 1
## 1 0.3333333 1 1 1 1
```

(e) Compute the marginal probability distribution for each node

```
unnormalised = array(0,dim = c(5,2),
                     dimnames=list(c("p(X1)","p(X2)","p(X3)",
                     "p(X4)","p(X5)",
                     c("0","1"))))
for(i in 1:5){
  unnormalised[i,] = forward_messages[i,]*backward_messages[,i]
```

```
}  
Z = rowSums(unnormalised)  
marginal_probability = unnormalised/Z  
marginal_probability
```

```
##           0           1  
## p(X1) 0.6666667 0.3333333  
## p(X2) 0.2444444 0.7555556  
## p(X3) 0.5735450 0.4264550  
## p(X4) 0.4852910 0.5147090  
## p(X5) 0.3570253 0.6429747
```