Finding trees in the network: Some Matlab programs and application in tumor pathways

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China; International Academy of Ecology and Environmental Sciences, Hong Kong
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 28 November 2015; Accepted 22 December 2015; Published online 1 June 2016

Abstract
Both DFS and Minty algorithms are used to find trees in a network (graph). In present article I present full Matlab codes of the two algorithms for using in the studies of network pharmacology. Trees are found in tumor pathways.

Keywords network; tree; Matlab; DFS; Minty.

1 Introduction
In the graph theory, a graph without any circuit is called acyclic graph. Connected acyclic graph is called tree (Zhang, 2012). A tree is called the spanning tree of a graph, if the tree contains all vertices of the graph. A connected graph must contain a spanning tree. These statements are true for networks also. DFS (Depth First Search; Tarjan, 1972) algorithm is used to obtain a tree from a network (graph). Minty’s algorithm (Minty, 1965) can be used to obtain all trees in a network (graph). In present article, I will present full Matlab codes of the two algorithms for potential application in the studies of network pharmacology.

2 Algorithms
Assume there are totally \( n \) nodes (vertices) in the network (graph), and adjacency matrix of the network is \( d=(d_{ij}), i,j=1,2,...,n \), where \( d_{ij}=d_{ji} \), \( d_{ii}=0 \), and if \( d_{ij}=1 \) or \( d_{ji}=1 \), there is a link (connection) between nodes \( i \) and \( j \).

2.1 DFS algorithm
The DFS algorithm is as follows (Tarjan, 1972; Zhang, 2012): First, change the adjacency matrix to Adjacency Vertex Listing. The ID number of starting node to be searched is 1. If \( T \) is the set of links (edges) on the tree \( k \) is the sequence number, \( B \) is the set of links not on the tree, \( v \) is the node being checked, \( w \) is the node to be checked, and \( n(i) \) is the ID number of each node, then

(1) Let \( v=1, k=1, j=1, n(1)=1 \).
(2) Search the incidence link that is not yet checked:

First, take the first link of \( v \), being not yet checked, and set it to be \((v, w)\). Reach the node \( w \) from this link.

The direction of the link \((v, w)\) is from \( v \) to \( w \). Return to (3).

If such a link was not found after each of the incidence links of \( v \) has been checked, return to (4).

(3) If \( w \) is the node being not yet visited (i.e., \( n(w) \) has not yet been determined), put the link \((v, w)\) to \( T \), and let \( v^\prime = w \), \( k = k + 1 \), \( n(w) = k \).

If \( w \) is the node that has been visited (i.e., \( n(w) \neq 0 \)), send the link \((v, w)\) into \( B \), return to the node \( v \), and let 
\( j = j + 1 \), return to (2).

(4) Determine the link \((u, v)\) that orients to node \( v \) in \( T \). Find out this link and return to node \( u \), let \( v = u \), and return to (2). If there is not such a link, terminate the calculation.

The Matlab codes for the DFS algorithm, DFS.m, are as follows

```
% DFS algorithm to obtain a tree in a network/graph.
function [tree,k,t1,t2,b1,b2,num]=DFS(d)
% d: adjacency matrix of the network; adjacency matrix is d=(dij)n*n, where n is the number of nodes in the network. dij=1 if
% vi and vj are adjacent, and dij=0, if vi and vj are not adjacent; i, j=1,2, ..., n.
% tree: string of a tree and all parameters and vectors.
% k: number of links on the tree; l: number of links not on the tree.
% t1[], b1[]: start nodes; t2[], b2[]: end nodes.
% t1[],t2[]: set of links on the tree; b1[],b2[]: set of links not on the tree.
% num[]: DFS labels of nodes.

n=size(d,1);
r=zeros(1,n);
r = sum(d);
e=max(r);
p=zeros(n,e);
for i=1:n
    m=0;
    for j=1:n
        if (d(i,j)~=0) m=m+1;p(i,m)=j; end
    end
    for j=1:n
        if (d(i,j)~=0) m=m+1;p(i,m)=j; end
    end
    num=zeros(1,n);
t1=zeros(1,n);
t2=zeros(1,n);
b1=zeros(1,n*e);
b2=zeros(1,n*e);
k=1; l=1; v=1; num(1)=1;
for i=2:n
    num(i)=0;
end
lab3=0;
while (n>0)
    s=r(v);
    while (n>0)
        lab2=0;
```
for i=1:s
if (p(v,i)==0) continue; end
w=p(v,i);
p(v,i)=0;
for j=1:r(w)
if (p(w,j)==v) p(w,j)=0; break; end
end
lab1=0;
if (num(w)==0)
t1(k)=v;
t2(k)=w;
k=k+1;
num(w)=k;
v=w;
lab1=1; break;
else
b1(l)=v;
b2(l)=w;
l=l+1;
lab2=1; break;
end; end
if (lab1==1) break;
elseif (lab2==1) continue; end
if (num(v)==1)
m=num(v)-1;
v=t1(m);
break;
end
lab3=1; break;
end
if (lab1==1) lab1=0; continue; end
if (lab3==1) break; end
end;
k=k-1;
l=l-1;
tree='A tree in the network/graph:
';
for i=1:k
tree=strcat(tree,'(',num2str(t1(i)),',',num2str(t2(i)),')');
if (i==k) tree=strcat(tree,''); end
end
tree=strcat(tree,'
DFS labels of nodes (num[]): 
');
for i=1:n
tree=strcat(tree,num2str(num(i)));
if (i==n) tree=strcat(tree,''); end
end
tree=strcat(tree,'\nStart nodes of the links on the tree (t1[])\n');
for i=1:k
    tree=strcat(tree,num2str(t1(i))); if (i\=k) tree=strcat(tree,');\n'; end
end

for i=1:k
    tree=strcat(tree,'\nEnd nodes of the links on the tree (t2[])\n');
    tree=strcat(tree,num2str(t2(i))); if (i\=k) tree=strcat(tree,');\n'; end
end

for i=1:l
    tree=strcat(tree,'\nStart nodes of the links not on the tree (b1[])\n');
    tree=strcat(tree,num2str(b1(i))); if (i\=l) tree=strcat(tree,');\n'; end
end

for i=1:l
    tree=strcat(tree,'\nEnd nodes of the links not on the tree (b2[])\n');
    tree=strcat(tree,num2str(b2(i))); if (i\=l) tree=strcat(tree,');\n'; end
end

2.2 Minty’s algorithm

Suppose an arbitrary link (edge) of a network (graph) \(X\) is \(e_i\). Divide all trees into two categories based on \(e_i\), in which a category contains \(e_i\) and another one does not contain \(e_i\). Find two subnetworks (subgraphs) \(X_1\) and \(X_2\) from \(X\), where adds \(e_i\) in \(X_1\), and eliminates \(e_i\) in \(X_2\). Every tree in \(X_1\) is added with \(e_i\), which forms the first category of trees in \(X\), and all trees in \(X_2\) belong to the second category of trees in \(X\). Choose another link (edge), repeat above procedures to get two subnetworks (subgraphs) from \(X_1\) and \(X_2\) respectively. In such a way, two new subnetworks (subgraphs) can be obtained each time. If the graph becomes a loop, then delete this subnetwork (subgraph). By removing all links (edges), all links (edges) of the subnetwork (subgraph) constitutes a tree. All trees are obtained after every subnetwork (subgraph) is handled (Minty, 1965; Chan et al., 1982; Zhang, 2012).

Chan et al. (1982) made a revision on Minty’s algorithm. The Matlab codes for the revised Minty algorithm, Minty.m, are as follows

```matlab
function trees=Minty(d)
% Revised Minty algorithm to obtain all trees in a network/graph.
% d: adjacency matrix of the network; Adjacency matrix is d=(dij)n*n, where n is the number of nodes in the network. dij=1 if vi and vj are adjacent, and dij=0, if vi and vj are not adjacent; i, j=1,2,..., n.
% trees: string of all trees
n=size(d,1);
e=sum(sum(d\=0))/2;
d1=zeros(1,e);
d2=zeros(1,e);
num=0;
for i=1:n-1
end
```
for j=i+1:n
    if (d(i,j)==0)
        num=num+1;
        d1(num)=i;
        d2(num)=j;
        end
    end
end

trees="; edge=zeros(1,e);
vmem=zeros(n*e,n);
emem=zeros(n*e,e);
tree=zeros(1,e);
vert=zeros(1,n);
for i=1:e
    edge(i)=1;
end
for i=1:n
    vert(i)=0;
end
k=1;
f=1;
s=0;
while (n>0)
    lab1=0; lab2=0;
    for j=1:e
        if (edge(j)==1) continue; end
        l=j;
        edge(j)=0;
        m=0;
        for i=1:e
            if (edge(i)==0) m=m+1; end
        end
        if (m>=(n-1))
            for i=1:e
                emem(f,i)=edge(i);
            end
            for i=1:n
                vmem(f,i)=vert(i);
            end
            f=f+1;
        end
    end
    edge(l)=-1;
    v1=d1(l);
    v2=d2(l);
    if (vert(v1)==0)
if (vert(v2)==0)
    vert(v1)=k;
    vert(v2)=k;
    k=k+1;
    lab1=1; break;
end
vert(v1)=vert(v2);
elseif (vert(v2)==0) vert(v2)=vert(v1);
else
    l=vert(v1);
    m=vert(v2);
    if ((l-m)==0) break; end
    if ((l-m)>0)
        t=m;
        m=l;
        l=t;
    end
    for i=1:n
        if ((vert(i)-m)==0) vert(i)=l; end
        if ((vert(i)-m)>0) vert(i)=vert(i)-1; end
    end
    k=k-1;
end;
for i=1:n
    if ((vert(i)-m)==0) lab2=1; break; end
end
if (lab2==1) break; end
s=s+1;
l=1;
for i=1:e
    if (edge(i)==-1)
        tree(l)=i;
        l=l+1;
    end; end
    trees=strcat(trees,'All links of tree No.',num2str(s),';
    for i=1:l-1
        trees=strcat(trees,'(',num2str(d1(tree(i))),',',num2str(d2(tree(i))),')');
    if (i==l-1) trees=strcat(trees,''); end
end
    trees=strcat(trees,'
    fprintf(trees)
end
if ((lab1==1) | (lab2==1)) continue; end
if (f==1) break; end
f=f-1;
for i=1:e; 
edge(i)=emem(f,i); 
end 
k=0; 
for i=1:n 
vert(i)=vmem(f,i); 
if (vmem(f,i)>=k) k=vmem(f,i); 
end 
k=k+1; 
end 

3 Application

Use DFS algorithm and the adjacency matrices of tumor pathways (Huang and Zhang, 2012; Li and Zhang, 2013; Zhang, 2016), the calculated tree in the p53 network is: (1,52),(52,4),(52,13),(52,15),(52,17),(52,19),(52,30),(52,48),(48,16),(16,18),(18,50),(50,20),(50,24),(24,47),(47,26),(47,32),(32,40),(40,42),(42,38),(38,41),(40,43),(47,33),(47,34),(47,35),(47,36),(47,37),(47,39),(47,44),(47,45),(47,46),(50,51),(51,49),(49,21),(49,23),(49,25),(49,27),(16,29),(29,31); for Ras tumor pathway, the calculated tree in the network is: (1,2),(2,3),(3,5),(5,4),(4,6),(4,8),(5,7),(5,9),(9,11),(11,13),(13,15),(15,17),(17,35),(35,33),(33,32),(32,31),(31,28),(28,26),(26,23),(23,21),(21,29),(29,30),(30,27),(32,34),(5,10),(10,12),(12,14),(12,19),(19,16),(16,18),(5,22),(22,20),(22,24),(5,25); for HGF pathway, the tree is: (1,2),(1,6),(6,8),(1,7), and for JNK tumor pathway, the searched tree is: (1,6),(6,5),(5,7),(7,9),(11,13),(13,8),(8,9),(13,10),(13,12),(13,14),(13,15),(13,24),(14,21),(21,26),(26,16),(26,22),(22,27),(27,28),(28,29),(29,30),(30,27),(32,34),(5,10),(10,12),(12,14),(12,19),(19,16),(16,18),(5,22),(22,20),(22,24),(5,25);

Use revised Minty algorithm and the adjacency matrix of p53 tumor pathway, the calculated trees (three trees are listed here) are as follows

Tree No.1:
(1,52),(2,5),(2,8),(2,10),(1,2),(2,14),(3,5),(4,5),(4,28),(4,52),(5,6),(5,7),(7,9),(11,13),(13,8),(8,9),(13,10),(13,12),(13,14),(13,15),(13,24),(21,26),(26,16),(26,22),(22,27),(27,28),(28,29),(29,30),(30,27),(32,34),(5,10),(10,12),(12,14),(12,19),(19,16),(16,18),(5,22),(22,20),(22,24),(5,25)

Tree No.2:
(1,52),(2,5),(2,8),(2,10),(1,2),(2,14),(3,5),(4,5),(4,28),(4,52),(5,6),(5,7),(7,9),(11,13),(13,8),(8,9),(13,10),(13,12),(13,14),(13,15),(13,24),(21,26),(26,16),(26,22),(22,27),(27,28),(28,29),(29,30),(30,27),(32,34),(5,10),(10,12),(12,14),(12,19),(19,16),(16,18),(5,22),(22,20),(22,24),(5,25)

Tree No.3:
(1,52),(2,5),(2,8),(2,10),(1,2),(2,14),(3,5),(4,5),(4,28),(4,52),(5,6),(5,7),(7,9),(11,13),(13,8),(8,9),(13,10),(13,12),(13,14),(13,15),(13,24),(21,26),(26,16),(26,22),(22,27),(27,28),(28,29),(29,30),(30,27),(32,34),(5,10),(10,12),(12,14),(12,19),(19,16),(16,18),(5,22),(22,20),(22,24),(5,25)
References