A Matlab algorithm for detection of protein complexes from multiple heterogeneous networks

WenJun Zhang, ShangHong Xin
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 12 October 2019; Accepted 18 December 2019; Published 1 June 2020

Abstract
In present article, we presented the Matlab algorithm of Ou-Yang’s model (Ou-Yang et al., 2017). It can be used to explore the shared clustering structure in PPI (protein-protein interaction) and DDI (domain-domain interaction) networks. A final matrix $H$ can be achieved using the algorithm. Protein $i$ belongs to complex $k$ if $H_{ik}=1$, otherwise $H_{ik}=0$, $i=1, 2, \ldots, N_1; k=1, 2, \ldots, K$, where $N_1$ is the number of proteins in PPI network, and $K$ is the number of complexes (clusters).

Keywords Matlab algorithm; Ou-Yang’s model; protein-protein interactions; protein complexes; heterogeneous networks.

1 Introduction
Biological systems at different levels are self-organizing systems (Zhang, 2013a, b, 2015, 2016, 2018). The organisms are survived by numerous protein-protein interactions (PPIs) in the cells. Proteins usually generate protein complexes to function (Huang et al., 2013; Zhao et al., 2014; Ou-Yang et al., 2017). Therefore, identification of protein complexes is a necessity. So far, numerous methods have been proposed in this aspect (Enright et al., 2002; Bader and Hogue, 2003; King et al., 2004; Adamscek et al., 2006; Li et al., 2010; Wang et al., 2010; Nepusz et al., 2012; Ji et al, 2014).

PPIs generally covers the physical interaction between specific protein domains (Wuchty, 2006. Ou-Yang et al., 2017). Since most proteins are multi-domain proteins, it is possible to develop algorithms that allow exploration of multiple between-node relationships in different networks (Greene et al., 2008; Zhang et al., 2012; Ou-Yang et al., 2013).

Based on previous studies, Ou-Yang et al. (2017) proposed a multi-network clustering (MNC) model to explore the shared clustering structure in PPI and DDI networks, in order to improve the accuracy of protein complex detection. Correspondingly, we here present the Matlab algorithm of Ou-Yang’s model (Ou-Yang et al., 2017).
2 Methods

2.1 Ou-Yang’s model

In Ou-Yang’s model, the networks are assumed to be collected from different but related fields, i.e., PPI network and DDI network. And it has a many-to-many (i.e., a protein may contain multiple domains) cross-field instance relationship. Given a PPI network and a DDI network, a model is used to describe the generation processes of two networks. Based on the domain-protein associations, the generation of PPI and DDI networks is assumed to be dominated by a shared clustering structure that describes the degree of proteins belonging to complexes. The protein complex detection finally becomes a parameter estimation problem (Ou-Yang et al., 2017).

According to Ou-Yang et al. (2017), given a PPI network $G_1$ with $N_1$ proteins, and a DDI network $G_2$ with $N_2$ domains, two nonnegative score matrices, $A^{(1)}_{N_1 \times N_1}$ and $A^{(2)}_{N_2 \times N_2}$, are the affinity/adjacency matrix of $G_1$ and $G_2$ respectively. The relationships between nodes in $G_1$ and nodes in $G_2$ may be many-to-many. The domain-protein associations are described by the domain-protein association matrix $FN_{N_2 \times N_1}$, where $F_{xi} = 1$ if protein $i$ in $G_1$ contains domain $x$ in $G_2$, otherwise $F_{xi} = 0$. The goal is to jointly find clustering structures in PPI network $G_1$ and DDI network $G_2$, and derive $H^{(m)}_{ik}$ (the weight of node $i$ in the predicted $k$-th cluster of $m$-th network) from each network $A^{(m)}$. A higher value of $H^{(m)}_{ik}$ means that node $i$ more likely belongs to cluster $k$ and vice versa. Here, $H^{(1)}=H, H^{(2)}=FH$, $H \in \mathbb{R}^{N_1 \times K}$, where $H$ is the protein-complex membership matrix.

Solve the following optimization problem:

$$
\min_{H, \lambda} - \sum_{i,j=1}^{N_1} A^{(1)}_{ij} \log(1 - \exp(-\sum_{k=1}^{K} H_{ik} H_{jk})) + \sum_{i,j=1}^{N_1} (1 - A^{(1)}_{ij}) \sum_{k=1}^{K} \lambda_k H_{ik} H_{jk} \\
- \sum_{x,y=1}^{N_2} F_{xy} \log(1 - \exp(-F \sum_{k=1}^{K} H_{ik} H_{jk} F_{xy})) + \sum_{x,y=1}^{N_2} (1 - F_{xy}) \lambda_k H_{ik} H_{jk} \\
+ \sum_{i=1}^{N_1} \sum_{k=1}^{K} \frac{H_{ik}^2}{2 \lambda_k} + \frac{\lambda_k}{2} \\
+ (a+1) \sum_{k=1}^{K} \log \lambda_k \\
H \geq 0
$$

(1)

in which

$$
\lambda_k \leftarrow \frac{(2b + \sum_{j=1}^{N_1} H_{jk}^2)}{(N_1 + 2a + 2)} \\
k = 1, 2, \ldots, K
$$

(2)

and

$$
H_{ik} \leftarrow H_{ik} / 2 + H_{ik} / 2 * (\sum_{j=1}^{N_1} A^{(1)}_{ij} H_{jk} / (1 - \exp(-H^{(1)}_{ij}))) + \sum_{x,y=1}^{N_2} (A^{(2)}_{xy} F_{xy} \sum_{j=1}^{N_1} H_{jk} F_{yj}) / (1 - \exp(-F \sum_{k=1}^{K} H_{ik} H_{jk} F_{xy})) \\
/ (\sum_{j=1}^{N_1} H_{jk} + \sum_{x,y=1}^{N_2} F_{xy} \sum_{j=1}^{N_1} H_{jk} F_{yj} + H_{ik} / (2 \lambda_k)) \\
i = 1, 2, \ldots, N_1; k = 1, 2, \ldots, K
$$

(3)

are alternatively changed and are used to minimize eq. (1) until the permitted iterative error of objective function is achieved. Initial $H$ (binary matrix) should be given before computation. In the initial $H$, $H_{ik}=1$, if
protein \( i \) is assigned to complex (cluster) \( k \), and \( H_{ik} = 0 \) otherwise, where \( i = 1, 2, \ldots, N_1; k = 1, 2, \ldots, K \). Initial \( H \)
is then positively perturbed with small positive and random values

\[
H \leftarrow H + \text{rand}(N_1,K)/10
\]

For the final \( H \), given a threshold \( \tau \). Protein \( i \) is assigned to complex \( k \) if \( H_{ik} \geq \tau \), i.e., let \( H_{ik} = 1 \), otherwise \( H_{ik} = 0 \) if \( H_{ik} < \tau \).

2.2 Matlab algorithm

The following is the full Matlab algorithm, PPI_DDI, for Ou-Yang’s model, using in the Matlab environment.

```matlab
a=input('Input parameter a (e.g., 2) = ');
b=input('Input parameter b (e.g., 0.25N1; e.g., for N1=100, b=250) = ');
sim=input('Input maximum number of iterations (e.g., 1000) = ');
err=input('Input permitted absolute error (e.g., 0.001) = ');
tao=input('Input threshold tao (Suggested: 0.3) = ');
pert=input('Input strength of random perturbation to H (Suggested: 0.1) = ');
strA1=input('Input the file name of A1 matrix (A1=(a1ij)N1\times N1): ','s');
strA2=input('Input the file name of A2 matrix (A2=(a2ij)N2\times N2): ','s');
strF=input('Input the file name of F matrix (F=(fij)N2\times N1): ','s');
strH=input('Input the file name of initial H matrix (H=(hij)N1\times K, where hij=1 if protein i is assigned to complex j, or else hij=0):','s');
H=xlsread(strH);
K=size(H,2);
A1=xlsread(strA1); A2=xlsread(strA2); F=xlsread(strF);
N1=size(A1,1);
N2=size(A2,1);
H0=H;
Hopt=H;
H=H+rand(N1,K)*pert;  %Positive random perturbation to H
objLast=1e+10;
sm=0;
while (sm<=sim)
  while (K>0)
    lamda=lamda_Update(H,a,b);
    \%H>=0
    if (sum(sum(H<0))>0)
      H=H_Update(H,lamda,A1,A2,F);
    else break;
    end
  end
  obj=objFun(H,lamda,A1,A2,F,a,b);
  if (obj<objLast)
    Hopt=H;
  end
  if (abs(objLast-obj)<err) break; end
```

IAEES www.iaeex.org
H=H_Update(H,lamda,A1,A2,F);
objLast=obj;
sm=sm+1;
end
Hopt(Hopt>=tao)=1;
Hopt(Hopt<tao)=0;
fprintf(['\n\nThe original matrix Hn']);
H0
fprintf(['\nThe optimal matrix Hn']);
Hopt
for k=1:K
fprintf(['\nThe proteins belonging to complex ' num2str(k) ' :n']);
for i=1:N1
if (Hopt(i,k)==1)
    fprintf([num2str(i) ',']);
end
end
end

function objFun=objFun(H,lamda,A1,A2,F,a,b)
N1=size(A1,1);
N2=size(A2,1);
K=size(H,2);
term1=0;
term2=0;
for i=1:N1;
    for j=1:N1;
        s=0;
        for k=1:K;
            s=s+H(i,k)*H(j,k);
        end;
        term1=term1+A1(i,j)*log(1-exp(-s));
        term2=term2+(1-A1(i,j))*s;
    end;
end
EX=F*H*H'*F';
term3=0;
term4=0;
for x=1:N2;
    for y=1:N2;
        term3=term3+A2(x,y)*log(1-exp(-EX(x,y)));
        term4=term4+(1-A2(x,y))*EX(x,y);
    end;
end
term5=0;

for i=1:N1;
for k=1:K;
term5=term5+H(i,k)^2/(2*lamda(k));
end
end
term6=0;
term7=0;
for k=1:K;
term6=term6+log(lamda(k));
term7=term7+b/lamda(k);
end
term8=term6*(a+1);
term6=term6*N1/2;
objFun=-term1+term2-term3+term4+term5+term6+term7+term8;

function lamda=lamda_Update(H,a,b)
N1=size(H,1);
K=size(H,2);
for k=1:K;
lamda(k)=(2*b+sum(H(:,k).^2))/(N1+2*a+2);
end

function H=H_Update(H,lamda,A1,A2,F)
N1=size(A1,1);
N2=size(A2,1);
K=size(H,2);
EX1=H*H';
EX2=-F*H*H'*F';
for i=1:N1;
for k=1:K;
sn1=0;
for j=1:N1;
sn1=sn1+A1(i,j)*H(j,k)/(1-exp(EX1(i,j)));
end
ss=0;
for x=1:N2;
for y=1:N2;
s2=0;
for j=1:N1;
s2=s2+H(j,k)*F(y,j);
end
ss=ss+F(x,i)*s2;
s1=A2(x,y)*F(x,i)*s2;
end
sn2=sn2+(1-exp(EX2(x,y)));
end
deno = sum(H(:,k)) + ss + Hi(i,k)/(2*lamda(k));
no = sn1 + sn2;
Hi(i,k) = Hi(i,k)/2 + Hi(i,k)/2*no/deno;
end
end

The executable GUI software (see supplementary material) of the algorithm above is partly indicated in Fig. 1.

![Fig. 1 The executable GUI software of the algorithm.](image_url)

**Acknowledgment**

We are thankful to the support of The National Key Research and Development Program of China (2017YFD0201204), and Discovery and Crucial Node Analysis of Important Biological and Social Networks (2015.6-2020.6), from Yangling Institute of Modern Agricultural Standardization.

**References**


