Core forging and local limit theorems for the k-core in random graphs

5/17/2017

```
#### Core Forging and local limit theorems for the k-core of random \
    graphs ####
#### Declaration of variables ####
# d the average degree in G(n,m)
# p the fixed point of equation (1.1)
# q from equation (1.2)
k,p,d,q=var('k','p','d','q')
assume(k,'integer')
assume(k>2,d>k,0<p<1,0<q<1)
# qbar as in equation (1.6)
# here we use the explicit definition of qbar in terms of p,q
qbar=var('qbar')
qbar=(k-1)*q/((1-p)*d)
#### The vectors/matrices for the proof of Proposition 2.8 ####
# Definition of vectors/matrices as in Figure 2.
# Sigma, L, T as in Figure 2.
SIGMA=matrix(SR,4,[[(1/d)*(1-p)^2,0,0,0]
                   ,[0,(1/d)*p*(1-p),0,0]
                   ,[0,0,(1/d)*p*(1-p)*(1+qbar*(d*p*(1-qbar)-(k-1))),0]
                   ,[0,0,0,(1/d)*p^2*(1-p*d/(1-q)+d*(p+(1-p)*qbar))]])
L=matrix(SR,[[1-p,0,0],
             [0,1-p,1-p],
             [p*(1-qbar),(k-1)/d,0],
             [0,0,p/(1-q)]])
T=matrix(SR,[[-1,-1,0,0],
             [1,0,0,0],
             [0,1,0,0],
             [0,0,-2,-1],
             [0,0,1,0],
             [0,0,1,0],
             [0,0,0,1]])
# The diagonal matrix diag(nu)^-1
N=matrix(SR,3,[[1/(1-p),0,0],
```

```
                [0,1/(p*q),0],
                [0,0,1/(p*(1-q))]])
# The diagonal matrix diag(mu)^-1
M=matrix(SR,4,[[1/((1-p)^2),0,0,0],
                [0,1/(p*(1-p)),0,0],
                [0,0,1/(p*(1-p)),0],
                [0,0,0,1/(p^2)]])
# Block matrix from the definition of Q^-1
# Top left block
TL=matrix(SR,3)
TL=(L.transpose())*(SIGMA.inverse())*L+N
# Top right block
TR=matrix(SR,3,4)
TR=-L.transpose()*SIGMA.inverse()
# Bottom left block
BL=matrix(SR,4,3)
BL=-SIGMA.inverse()*L
# Bottom right block
BR=matrix(SR,4)
BR=SIGMA.inverse()-d/2*M
# Block matrix
Blockmatrix=matrix(SR,7,7)
Blockmatrix=block_matrix([[TL,TR],[BL,BR]])
# Definition of Q^-1
CovmatrixQInv=matrix(SR,4)
CovmatrixQInv=T.transpose()*Blockmatrix*T
# Verification of equation (5.32)
print 'Det(Q^-1)*Det(SIGMA)=', factor(CovmatrixQInv.determinant()*\
    SIGMA.determinant())
```

```
Det(Q^-1)*Det(SIGMA)=
1/2*(k*q - q - 1)^3/(d^2*p*(q - 1)*q)
```

```
#### Local limit theorems ####
# Here we determine explicit formulas for the entries of Q as in \
    Theorem 2.5.
# Definition of the covariance matrix Q
CovmatrixQ=matrix(SR,4)
CovmatrixQ=CovmatrixQInv.inverse()
# Entries of Q
for i in [0..3]:
    for j in [0..3]:
        print 'CovmatrixQ[',i+1,',',j+1,']=',
        ((k*q-q-1)^2*CovmatrixQ[i,j]).simplify_full()
```

```
CovmatrixQ[ 1 , 1 ]=
-((d*k^2 - 2*d*k + d)*p^2*q^4 + (2*(d^2*k - d^2)*p^3 - (2*d*k^2 - d^2 + (d^2 - 2*d)*k)*p^2
+ (d*k^2 - 2*d*k + d)*p)*q^3 - d*p*q + ((d^3 + 2*d^2)*p^4 - (d^3 + 2*(d^2 + 2*d)*k -
4*d)*p^3 + ((d + 2)*k^2 - d^2 + 2*(d^2 - 2)*k + 2)*p^2 - (d*k^2 - 2*d*k + d)*p)*q^2)/d
```

CovmatrixQ[1 , 2]=
(k^2 - 2*k + 1)*p^2*q^4 + (2*(d*k - d)*p^3 - ((d - 2)*k + 2*k^2 - d)*p^2 + (k^2 - 2*k + 1)*p)*q^3 + ((d^2 + 2*d)*p^4 - (d^2 + 2*(d + 1)*k - 2)*p^3 + ((2*d + 1)*k + k^2 - d - 1)*p^2 - (k^2 - k)*p)*q^2 + (d*p^3 - (d + k)*p^2 + (k - 1)*p)*q
CovmatrixQ[1 , 3]=
-((2*(d*k - d)*p^4 + 2*((d + 2)*k - k^2 - d - 1)*p^3 - 3*(d*k - d)*p^2 + ((d - 2)*k + k^2 - d + 1)*p)*q^2 + (2*(d^2 + d)*p^4 - (3*d^2 + 2*(d + 1)*k + 2*d - 2)*p^3 + (d^2 + (3*d + 2)*k - 2)*p^2 - ((d + 1)*k - 1)*p)*q)/d
CovmatrixQ[1 , 4]=
2*(((d*k - d)*p^4 + ((d + 2)*k - k^2 - d - 1)*p^3)*q^2 + ((d^2 + d)*p^4 - (d^2 + (d + 1)*k - 1)*p^3 + (d*k - d)*p^2)*q)/d
CovmatrixQ[2 , 1]=
(k^2 - 2*k + 1)*p^2*q^4 + (2*(d*k - d)*p^3 - ((d - 2)*k + 2*k^2 - d)*p^2 + (k^2 - 2*k + 1)*p)*q^3 + ((d^2 + 2*d)*p^4 - (d^2 + 2*(d + 1)*k - 2)*p^3 + ((2*d + 1)*k + k^2 - d - 1)*p^2 - (k^2 - k)*p)*q^2 + (d*p^3 - (d + k)*p^2 + (k - 1)*p)*q
CovmatrixQ[2 , 2]=
-(k^2 - 2*k + 1)*p^2*q^4 - (2*(d*k - d)*p^3 - ((d - 2)*k + 2*k^2 - d)*p^2 + (k^2 - 2*k + 1)*p)*q^3 - ((d^2 + 2*d)*p^4 - (d^2 + 2*d*k)*p^3 + (2*(d + 1)*k + k^2 - d - 2)*p^2 - (k^2 - 1)*p)*q^2 - p^2 - (2*d*p^3 - 2*(d + k)*p^2 + (2*k - 1)*p)*q + p
CovmatrixQ[2 , 3]=
2*p^3 + (2*(k - 1)*p^4 + 2*(k - 1)*p^3 - 3*(k - 1)*p^2 + (k - 1)*p)*q^2 - 3*p^2 + (2*(d + 1)*p^4 - (3*d + 2*k + 2)*p^3 + (d + 3*k)*p^2 - k*p)*q + p
CovmatrixQ[2 , 4]=
-2*p^3 - 2*((k - 1)*p^4 + (k - 1)*p^3)*q^2 + 2*p^2 - 2*((d + 1)*p^4 - (d + k)*p^3 + (k - 1)*p^2)*q
CovmatrixQ[3 , 1]=
-((2*(d*k - d)*p^4 + 2*((d + 2)*k - k^2 - d - 1)*p^3 - 3*(d*k - d)*p^2 + ((d - 2)*k + k^2 - d + 1)*p)*q^2 + (2*(d^2 + d)*p^4 - (3*d^2 + 2*(d + 1)*k + 2*d - 2)*p^3 + (d^2 + (3*d + 2)*k - 2)*p^2 - ((d + 1)*k - 1)*p)*q)/d
CovmatrixQ[3 , 2]=
2*p^3 + (2*(k - 1)*p^4 + 2*(k - 1)*p^3 - 3*(k - 1)*p^2 + (k - 1)*p)*q^2 - 3*p^2 + (2*(d + 1)*p^4 - (3*d + 2*k + 2)*p^3 + (d + 3*k)*p^2 - k*p)*q + p
CovmatrixQ[3 , 3]=
-(2*(2*d + 1)*p^4 - 4*(2*d + 1)*p^3 + (5*d + 3)*p^2 + (2*(k^2 - 2*k + 1)*p^4 - 2*(k^2 - 2*k + 1)*p^2 + (k^2 - 2*k + 1)*p)*q^2 - (d + 1)*p + (4*(k - 1)*p^4 - 4*(k - 1)*p^3 + (k - 1)*p^2)*q)/d
CovmatrixQ[3 , 4]=
2*((k^2 - 2*k + 1)*p^4*q^2 + (2*d + 1)*p^4 - (3*d + 1)*p^3 + d*p^2 + (2*(k - 1)*p^4 - (k - 1)*p^3)*q)/d
CovmatrixQ[4 , 1]=
2*(((d*k - d)*p^4 + ((d + 2)*k - k^2 - d - 1)*p^3)*q^2 + ((d^2 + d)*p^4 - (d^2 + (d + 1)*k - 1)*p^3 + (d*k - d)*p^2)*q)/d
CovmatrixQ[4 , 2]=
-2*p^3 - 2*((k - 1)*p^4 + (k - 1)*p^3)*q^2 + 2*p^2 - 2*((d + 1)*p^4 - (d + k)*p^3 + (k - 1)*p^2)*q
CovmatrixQ[4 , 3]=
2*((k^2 - 2*k + 1)*p^4*q^2 + (2*d + 1)*p^4 - (3*d + 1)*p^3 + d*p^2 + (2*(k - 1)*p^4 - (k -

1)*p^3)*q)/d
CovmatrixQ[4 , 4]=
-2*(2*(k - 1)*p^4*q + (2*d + 1)*p^4 - 2*d*p^3 + ((k^2 - 2*k + 1)*p^4 + (k^2 - 2*k + 1)*p^2)*q^2 - p^2)/d