HW3Visualizations

December 15, 2017

0.1 ECE 289: Homework 3

0.1.1 Part (A): Networks Visualization

In [25]:
```
%%time
#import useful packages, all of them are important but not necessarily used in this cod
#enable inline plotting in Python Notebook
#supress warnings

%pylab inline
import networkx as nx
import numpy as np
import matplotlib
import scipy
import warnings
warnings.filterwarnings('ignore')
import time

# import required packages
# load results from HW1 to use. Otherwise, execute code from HW1 to produce results aga
import ast
import re
```

```
Populating the interactive namespace from numpy and matplotlib
CPU times: user 2.33 s, sys: 0 ns, total: 2.33 s
Wall time: 3.02 s
```

```
/opt/conda/envs/python2/lib/python2.7/site-packages/IPython/core/magics/pylab.py:161: UserWarnin
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

get fb data and build graph

In [23]:
```
with open("fbdegDist.txt","r") as myfile:
    fbdd="".join(line.rstrip() for line in myfile)
fbddist=ast.literal_eval(fbdd)
```

```
        with open("fbbetCen.txt","r") as myfile:
            fbbc="".join(line.rstrip() for line in myfile)
        fbbetCent=ast.literal_eval(fbbc)


        #load the network after converting into text file
        file_name="../ece289_public/facebook_combined.txt"
        #convert the information in the text file into a graph, find no. of edges & nodes in th
        g1=nx.read_edgelist(file_name,create_using=nx.Graph(),nodetype=int)
        node, edge=g1.order(),g1.size()
        print "No. of nodes are=",node
        print "No. of edges are=",edge
        nx.set_node_attributes(g1, fbbetCent,"betCen")
        nx.set_node_attributes(g1, fbddist,"deg")
        nx.write_gml(g1,"fb.gml")
        print "facebook gml created"
```

```
No. of nodes are= 4039
No. of edges are= 88234
facebook gml created
```

get Enron data and build graph

```
In [24]: with open("endegDist.txt","r") as myfile:
             endd="".join(line.rstrip() for line in myfile)
         enddist=ast.literal_eval(endd)
         #myfile = open("enbetCen.txt")
         #txt = myfile.read()
         #with open("enbetCen.txt","r") as myfile:
         #    enbc="".join(line.rstrip() for line in myfile)
         #import string
         #enbc = txt.translate(string.maketrans('', ''), 'dpIFs')
         #enbc="".join(line.rstrip() for line in myfile)
         #enbetCent=ast.literal_eval(enbc)
         #%store enbc > enbc.txt
         enbc={}
         with open('enbc.txt', 'r') as f:
             count = 0
             by1=0
             for line in f:
                 count+=1
                 if count % 2 == 0: #this is the remainder operator
                     #print(line)
                     enbc[by1]=float(line.strip('\n'))
                     by1+=1

         file_name="../ece289_public/email-Enron.txt"
```

```
        #convert the information in the text file into a graph, find no. of edges & nodes in th
        g2=nx.read_edgelist(file_name,create_using=nx.Graph(),nodetype=int)
        node, edge=g2.order(),g2.size()
        print "No. of nodes are=",node
        print "No. of edges are=",edge
        graphs = list(nx.connected_component_subgraphs(g2, copy=True))
        # the connected component with the most nodes
        graph_max = sorted([(len(gn.nodes()), gn) for gn in graphs], key=lambda x: x[0], revers
        print nx.info(graph_max)
        g22=graph_max
        nx.set_node_attributes(g22, enbc,"betCen")
        nx.set_node_attributes(g22, enddist,"deg")
        nx.write_gml(g22,"en.gml")
        print "enron gml created"
```

```
No. of nodes are= 36692
No. of edges are= 183831
Name:
Type: Graph
Number of nodes: 33696
Number of edges: 180811
Average degree:  10.7319
enron gml created
```

get Erdos graph and build graph + some data calcs

```
In [27]: %%time
         # build Collaboration Network
         # undirected network
         g3 = nx.Graph()

         # add Paul Erdos into our network at first
         dict_authors = {}
         dict_authors['Paul Erdos'] = 0
         labelmap={}
         labelmap[0]='Paul Erdos'
         g3.add_node(0)
         g3.node[0]['author'] = 'Paul Erdos'

         # add the authors with Erdos number 1 and 2 from file
         line_count = 1
         skip_line = 24
         skip_space = 1

         is_new = False
         author = ""
         coauthor = ""
```

```
index = 1
ind_author = 1
ind_coauthor = 1

def parseLine(l, start):
    end = start
    while end < len(l) - 1 and not (l[end] == ' ' and l[end + 1] == ' '):
        end += 1
    return l[start:end]

def addAuthor(auth, ind):
    if auth in dict_authors:
        return ind
    dict_authors[auth] = ind
    #edit
    labelmap[ind]=auth
    #edit
    return ind + 1

for l in open("../ece289_public/Erdos.html"):
    if line_count >= skip_line:
        if l == '\n':
            is_new = True
        elif is_new:
            author = parseLine(l, 0)
            index = addAuthor(author, index)
            ind_author = dict_authors[author]
            g3.add_edge(0, ind_author)
            g3.node[ind_author]['author'] = author
            is_new = False
        elif l == '</pre>':
            break
        else:
            coauthor = parseLine(l, skip_space)
            index = addAuthor(coauthor, index)
            ind_coauthor = dict_authors[coauthor]
            g3.add_edge(ind_author, ind_coauthor)
            g3.node[ind_coauthor]['author'] = coauthor
    line_count += 1

print nx.info(g3)

node=g3.order()
deg=g3.degree()
d=list(np.zeros((node), dtype=np.int))
temp=list(np.zeros((node), dtype=np.int))
for k in range (node):
    d[k]= deg[k]
```

```
        for k in range (node):
            temp[k]=k
        degList=dict(zip(temp,d))
        for k in g3.nodes():
            g3.node[k]['deg'] = deg[k]
        deg_values = sorted(set(degList.values()))
```

```
Name:
Type: Graph
Number of nodes: 11524
Number of edges: 18504
Average degree:   3.2114
CPU times: user 192 ms, sys: 0 ns, total: 192 ms
Wall time: 192 ms
```

Erdos betweeness centrality calc

```
In [28]: %%time
         from multiprocessing import Pool
         import time
         import itertools

         G=g3

         def chunks(l, n):
             """Divide a list of nodes `l` in `n` chunks"""
             l_c = iter(l)
             while 1:
                 x = tuple(itertools.islice(l_c, n))
                 if not x:
                     return
                 yield x


         def _betmap(G_normalized_weight_sources_tuple):
             """Pool for multiprocess only accepts functions with one argument.
             This function uses a tuple as its only argument. We use a named tuple for
             python 3 compatibility, and then unpack it when we send it to
             `betweenness_centrality_source`
             """
             return nx.betweenness_centrality_subset(*G_normalized_weight_sources_tuple)


         def betweenness_centrality_parallel(G, processes=4):
             """Parallel betweenness centrality  function"""
             p = Pool(processes=processes)
             node_divisor = len(p._pool)*4
```

```
            node_chunks = list(chunks(G.nodes(), int(G.order()/node_divisor)))
            #print node_chunks
            num_chunks = len(node_chunks)


            bt_sc = p.map(_betmap,
                          zip([G]*num_chunks,
                              node_chunks,
                              [list(G)]*num_chunks,
                              [True]*num_chunks,
                              [None]*num_chunks))


            #print bt_sc

            # Reduce the partial solutions
            bt_c = bt_sc[0]
            for bt in bt_sc[1:]:
                for n in bt:
                    bt_c[n] += bt[n]
            return bt_c

        print("")
        print("Computing betweenness centrality for:")
        print(nx.info(G))
        print("\tParallel version")
        start = time.time()
        #G = graph_max
        #node_chunks = list(chunks(G.nodes(), int(G.order()/4)))
        #G = graph_max
        #print "SOURCE:", nx.betweenness_centrality_source(graph_max,True,False,list([1,2]))
        #bt = nx.betweenness_centrality_subset(G,[0],list(G),True,None)
        bt = betweenness_centrality_parallel(G)
        #bt = nx.betweenness_centrality(graph_max)
        print("\t\tTime: %.4F" % (time.time()-start))
        print("\t\tBetweenness centrality for node 0: %.5f" % (bt[0]))

        #print("\tNon-Parallel version")
        #start = time.time()
        #bt = nx.betweenness_centrality(G)
        #print("\t\tTime: %.4F seconds" % (time.time()-start))
        #print("\t\tBetweenness centrality for node 0: %.5f" % (bt[0]))

        print("")


Computing betweenness centrality for:
Name:
```

```
Type: Graph
Number of nodes: 11524
Number of edges: 18504
Average degree:   3.2114
        Parallel version
                Time: 263.6883
                Betweenness centrality for node 0: 0.75593

CPU times: user 1.01 s, sys: 0 ns, total: 1.01 s
Wall time: 4min 23s
```

```
In [31]: #%store bt > betCen.txt

         nx.set_node_attributes(g3, bt,"betCen")
         nx.set_node_attributes(g3, degList,"deg")
         #needed to make png with authornames instead of node ids
         g33=nx.relabel_nodes(g3,labelmap)


         #nx.write_gml(g33,"erdos.gml")
         #print "erdos gml created"
```

build cit-network graph

```
In [29]: %%time
         # load the network
         file_name = "../ece289_public/Cit-HepTh.txt"
         # it's a directed graph, so we should use nx.DiGraph to read
         g4 = nx.read_edgelist(file_name, create_using=nx.DiGraph(), nodetype=int)
         #print nx.info(g4)
         og4=g4
         g44=g4.to_undirected()
         connected=nx.is_connected(g44)
         graphs = list(nx.connected_component_subgraphs(g44, copy=True))
         # the connected component with the most nodes
         graph_max = sorted([(len(gn.nodes()), gn) for gn in graphs], key=lambda x: x[0], revers
         print nx.info(graph_max)
         g44=graph_max

Name:
Type: Graph
Number of nodes: 27400
Number of edges: 352059
Average degree:  25.6977
CPU times: user 12.6 s, sys: 52 ms, total: 12.7 s
Wall time: 12.4 s
```

0.2 Use network datasets from the Public Data Folder.

Please DO NOT copy dataset in your directory (or anywhere else!) because it will certainly result in 'timeout error' while submission.

Now that we are familiar with Gephi and its usage, we will explore some built in tools from Gephi to improve our visualizations further. You will need results from the HW1. Only use results calculated using networkX here. This can be done by adding your results for node degree, centrality etc. as an attribute to the

In this task, we will use Filter tool in Gephi to threshold available network data, using various properties.

Visualise Facebook Network, Enron Emails Network, Collaboration (Erdos) Network applying following thesholds. Make sure to have all the visualizations labelled with appropriate node labels. This is a quiet open ended question, as you have a lot of scope to make your visualizations better trying different layouts, colors etc. So, turn in the best visualization that you get in each case. You should attach an image(.png, .jpg) for each visualization here in Notebook itself. Also, make sure that it is well readable.

(1) Top ~50% nodes, thrsholded by Node Degree

(2) Top ~10% nodes, thrsholded by Node Degree

(3) Top ~5% nodes, thrsholded by Node Degree

(4) Top ~1% nodes, thrsholded by Node Degree

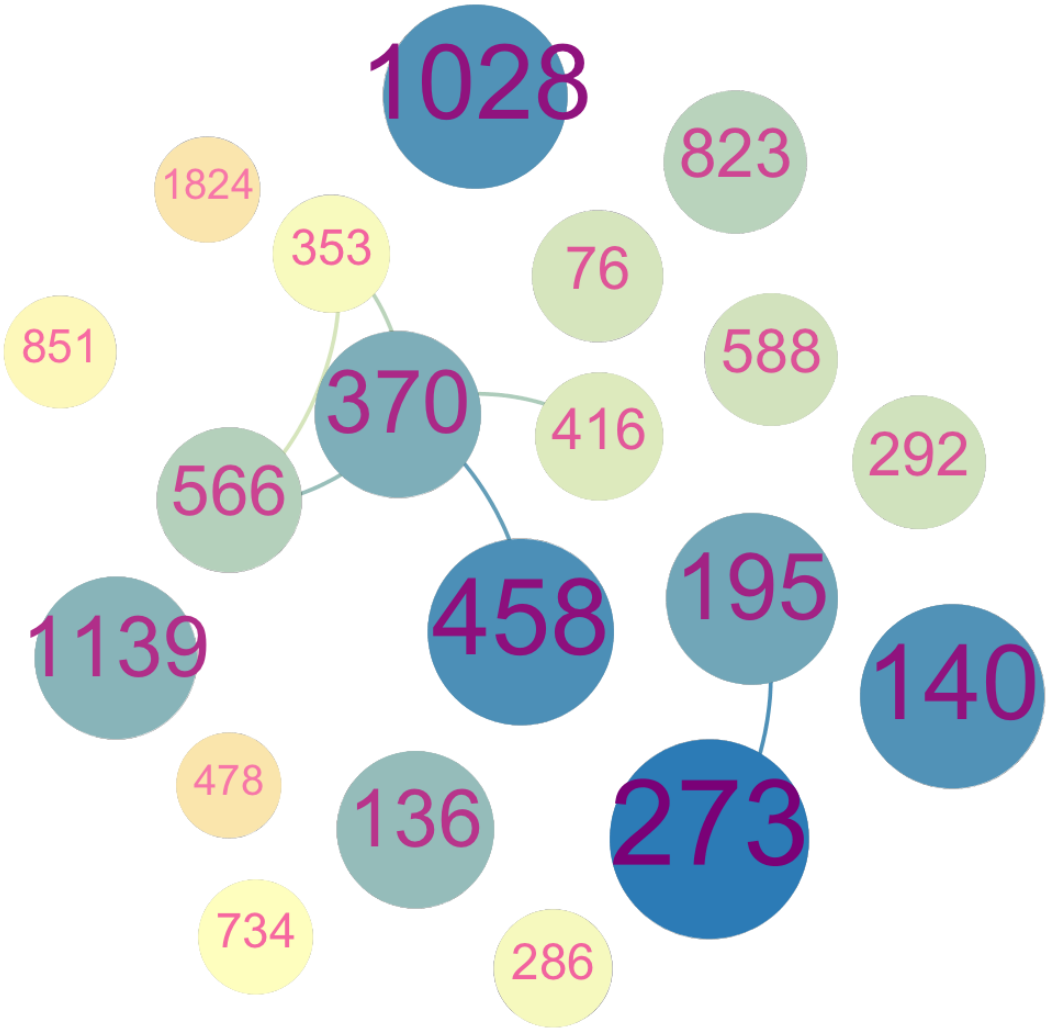
(5) Top ~50% nodes, thrsholded by Betweeness Centrality

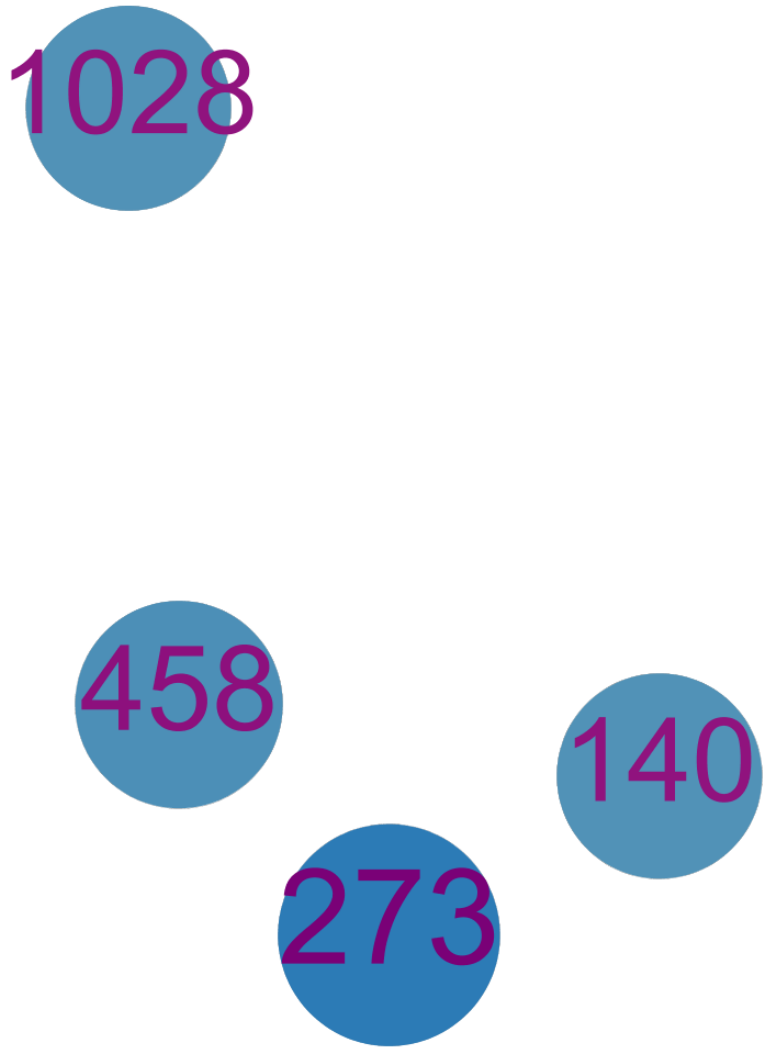
(6) Top ~10% nodes, thrsholded by Betweeness Centrality

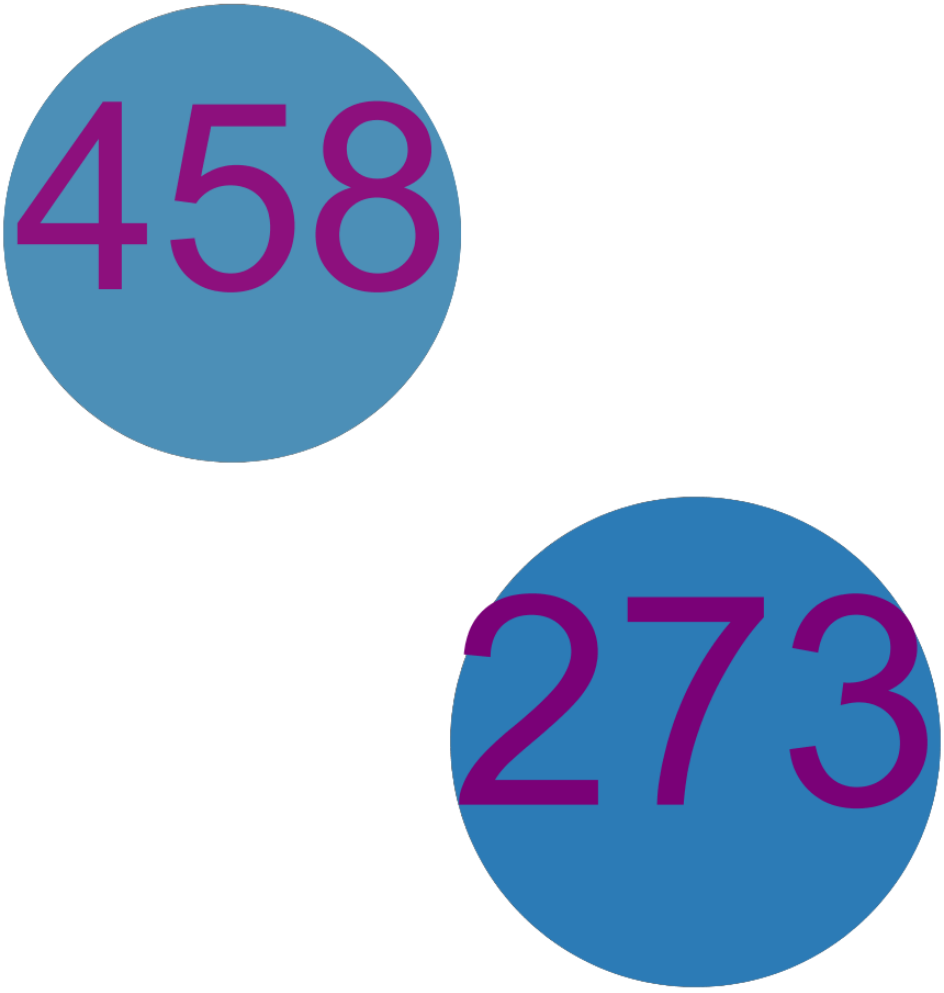
(7) Top ~5% nodes, thrsholded by Betweeness Centrality

(8) Top ~1% nodes, thrsholded by Betweeness Centrality facebook degree filters

```
In [7]: # your response images here
        from IPython.display import Image
        from IPython.display import display

        w=Image(filename='fbdeg50.png')
        x=Image(filename='fbdeg10.png')
        y=Image(filename='fbdeg05.png')
        z=Image(filename='fbdeg01.png')

        display(w, x, y, z)
```

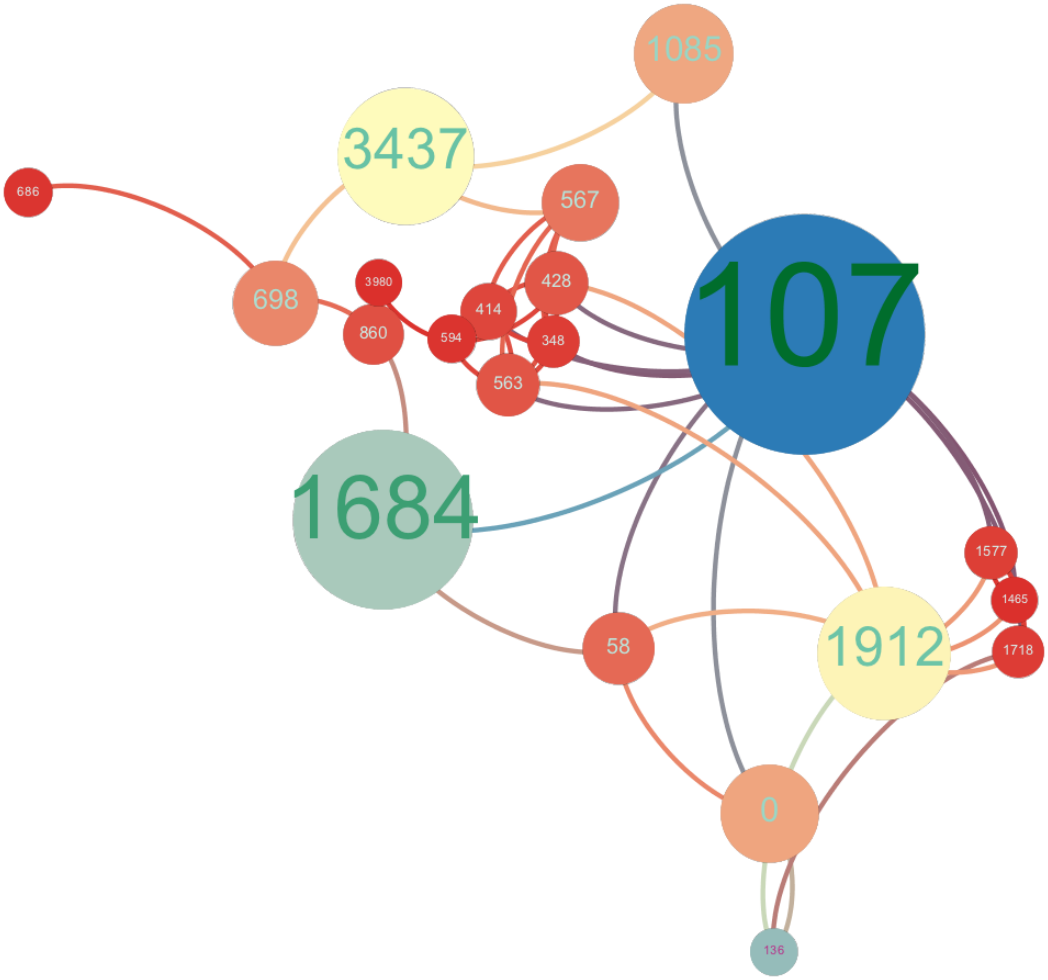

1028
823
1824
353
76
851
588
370
416
292
566
458
195
1139
140
478
136
273
734
286

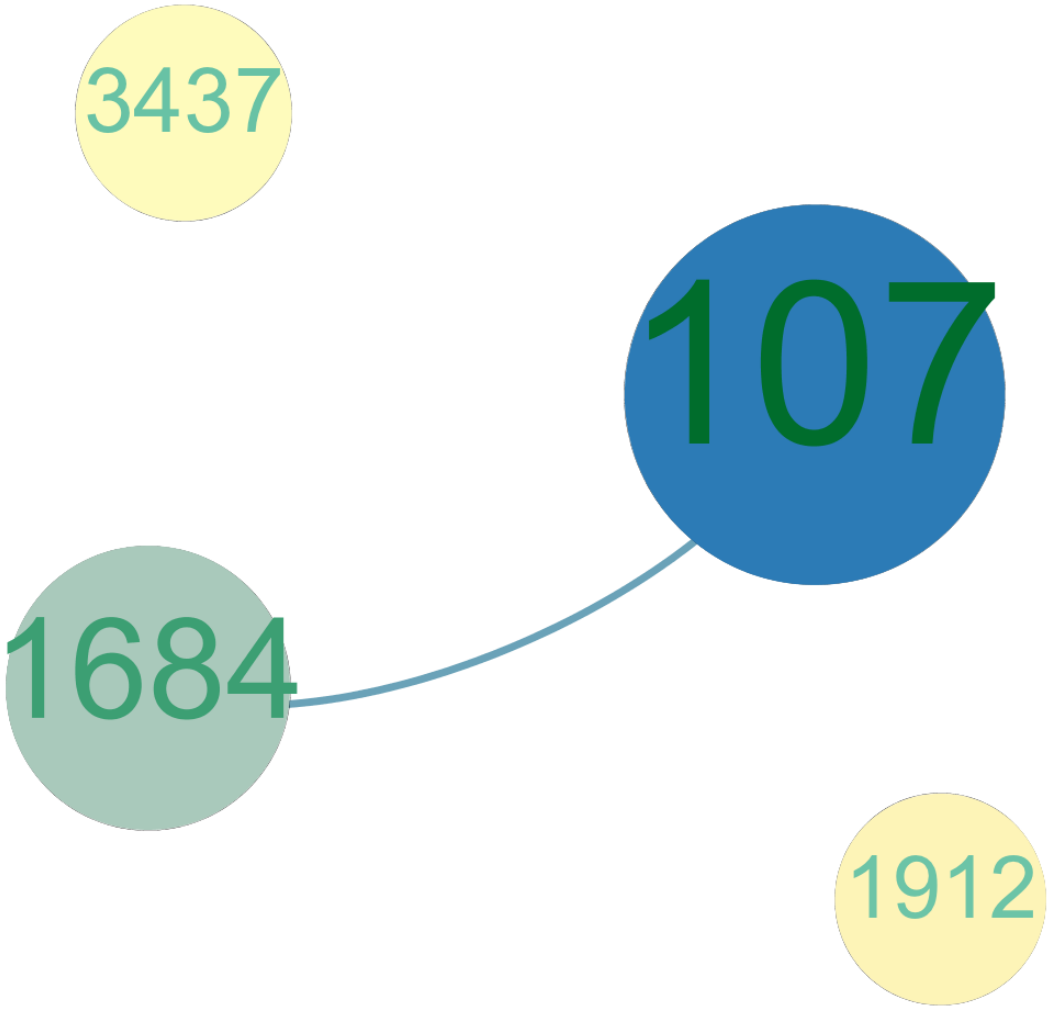

1028
458
140
273

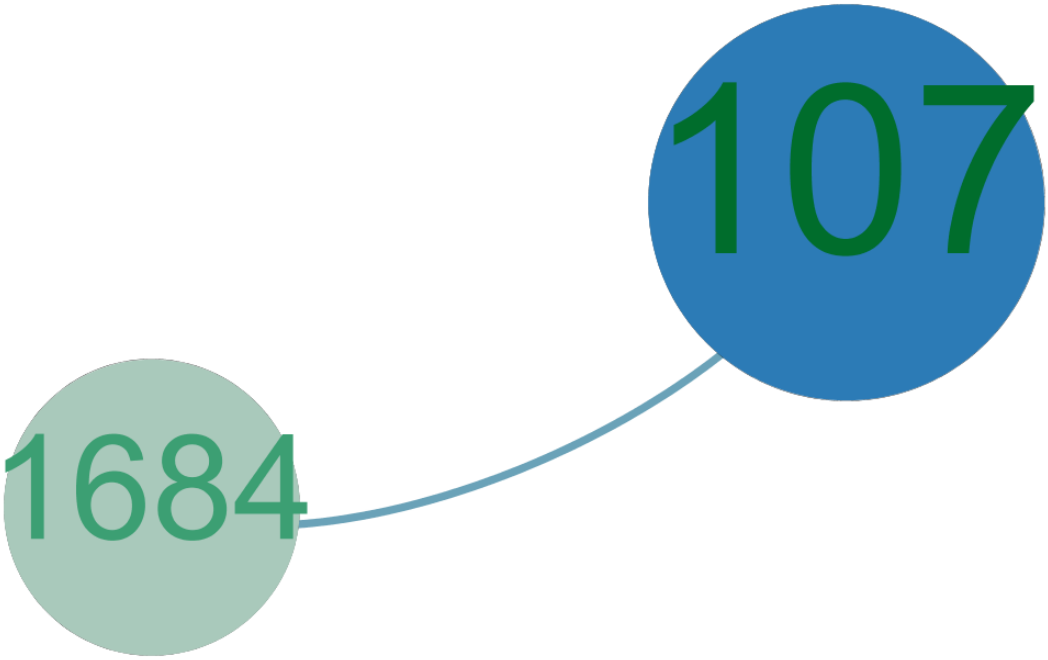

458
273

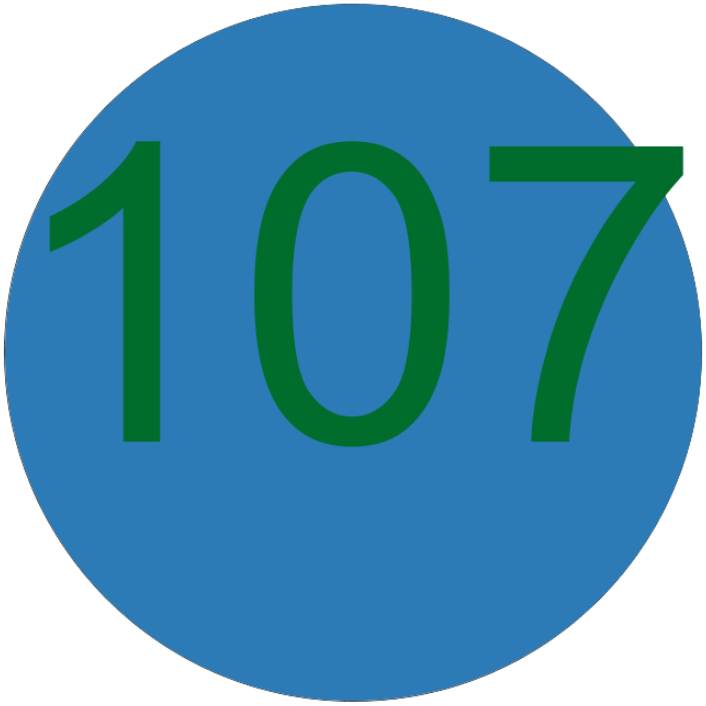
facebook betweenness centrality filters

```
In [8]: from IPython.display import Image
        from IPython.display import display
        ww=Image(filename='fbbetCen50.png')
        xx=Image(filename='fbbetCen10.png')
        yy=Image(filename='fbbetCent05.png')
        zz=Image(filename='fbbetCent01.png')

        display(ww, xx, yy, zz)
```


1085
3437
686
567
698
3980
428
107
414
860
594
348
563
1684
1577
1465
58
1912
1718
0
136

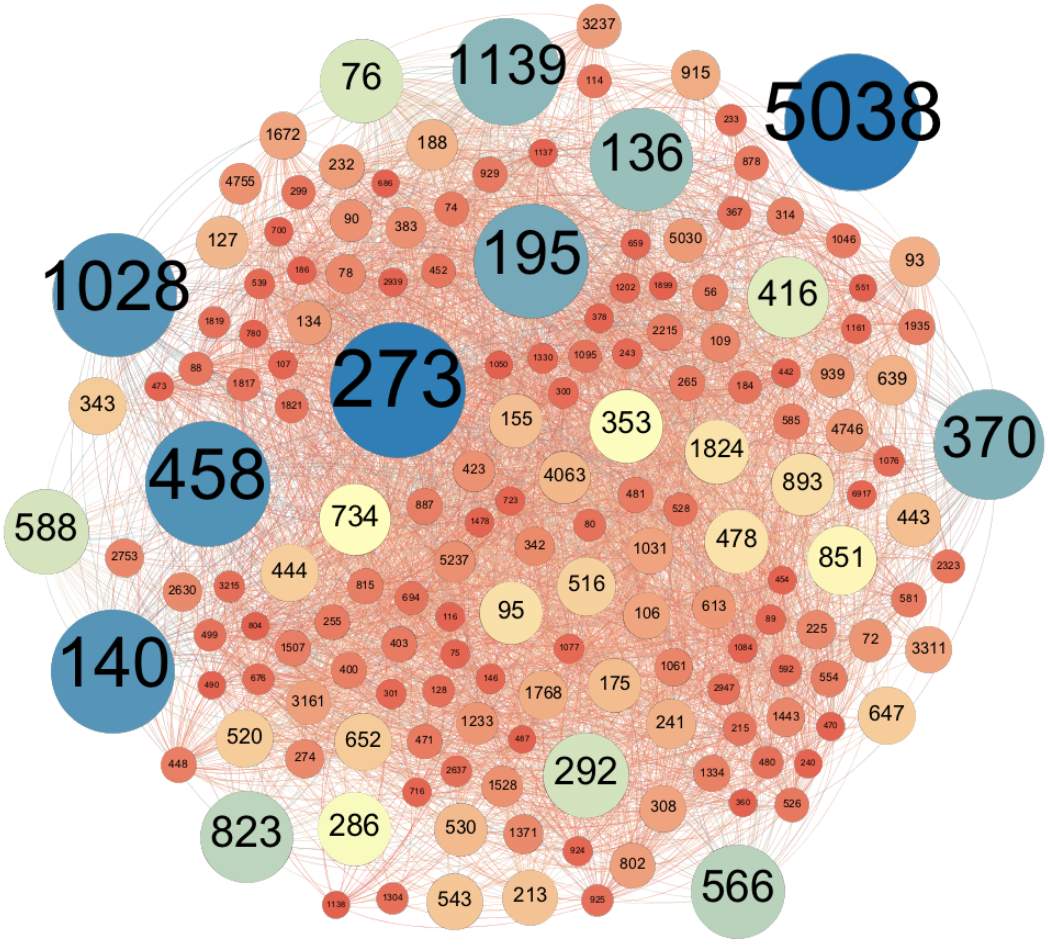

3437
107
1684
1912

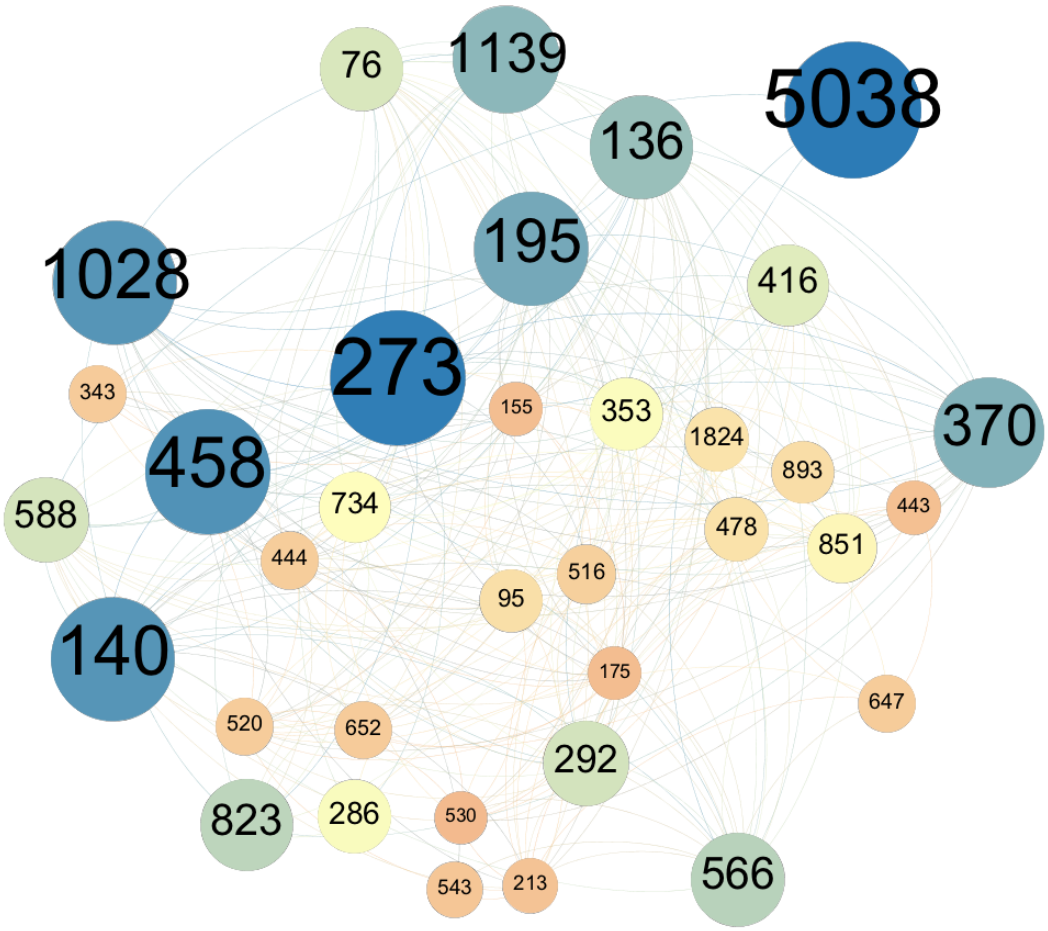

107
1684

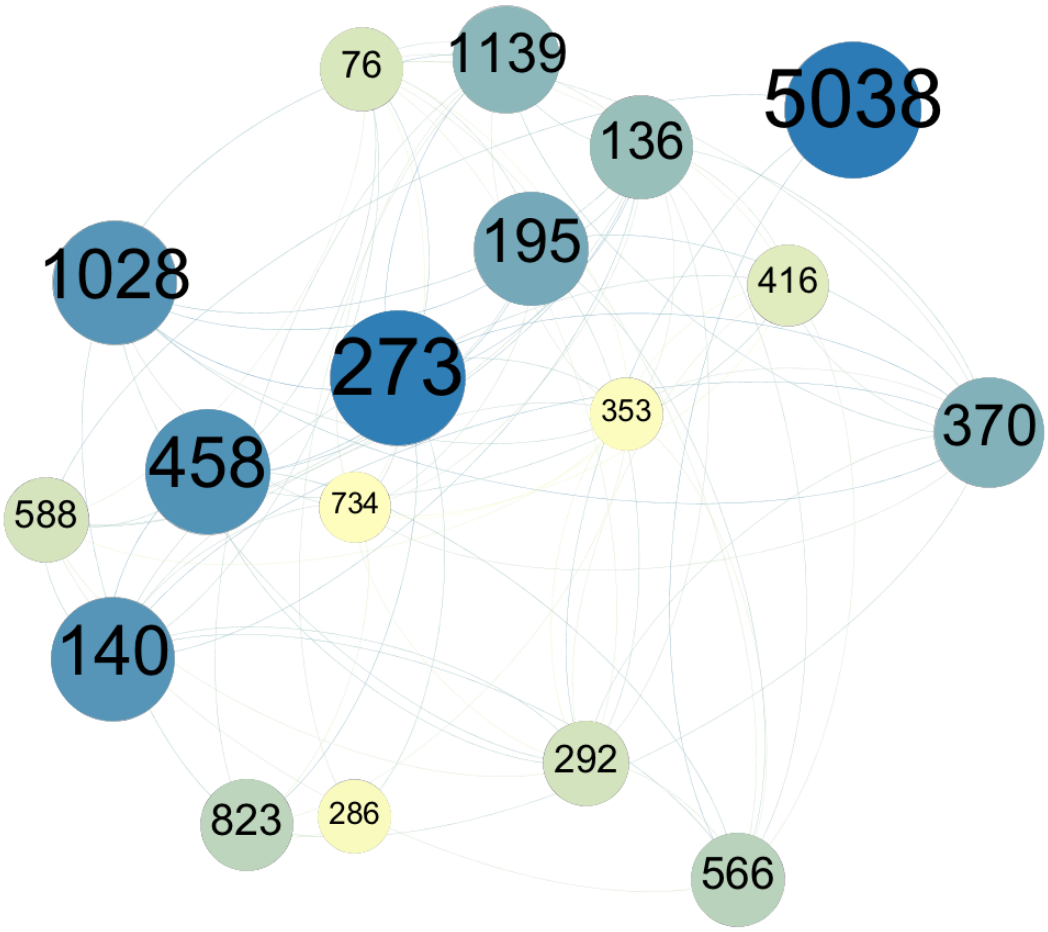


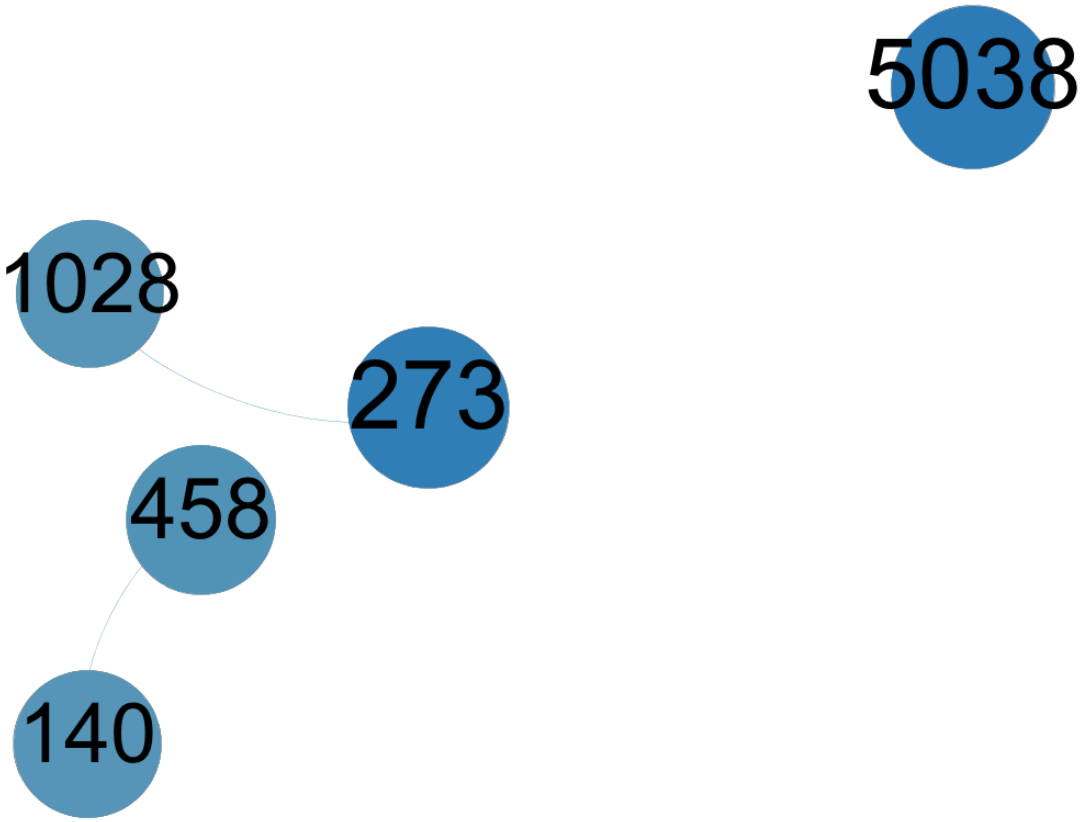
Enron degree filters

```
In [10]: from IPython.display import Image
         from IPython.display import display

         a=Image(filename='endeg50.png')
         b=Image(filename='endeg10.png')
         c=Image(filename='endeg05.png')
         d=Image(filename='endeg01.png')

         display(a,b,c,d)
```


3237
76
1139
915
5038
114
1672
188
136
233
232
878
1137
4755
929
686
299
74
90
383
367
314
127
700
195
659
5030
1046
1028
539
186
78
2939
452
1202
1899
56
416
93
551
1819
780
134
378
2215
109
1161
1935
273
1050
1330
1095
243
88
107
265
184
442
939
639
473
1817
343
300
155
353
585
4746
370
1821
1824
1076
423
4063
481
893
458
6917
887
723
528
443
734
1478
80
588
342
1031
478
851
2753
5237
444
516
2323
2630
3215
815
694
454
95
106
613
581
116
89
225
140
499
804
255
403
1077
1084
72
3311
1507
75
1061
592
400
146
1768
175
2947
554
490
676
301
128
241
1443
647
3161
1233
215
520
652
471
487
470
448
274
2637
1528
292
1334
480
240
716
308
360
526
823
286
530
1371
924
802
566
1138
1304
543
213
925

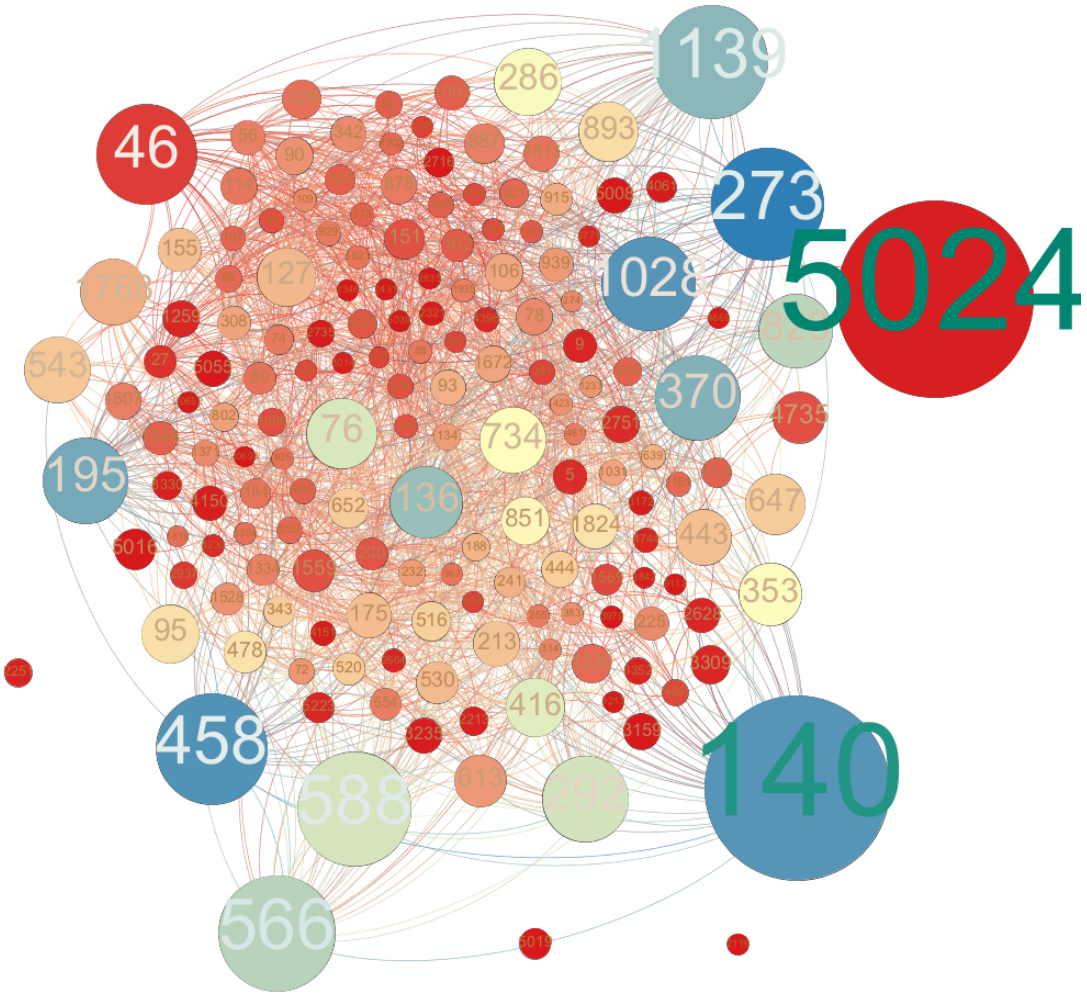

76
1139
5038
136
195
1028
416
273
343
155
353
1824
370
458
893
588
734
443
478
851
444
516
95
140
175
647
520
652
292
823
286
530
566
543
213

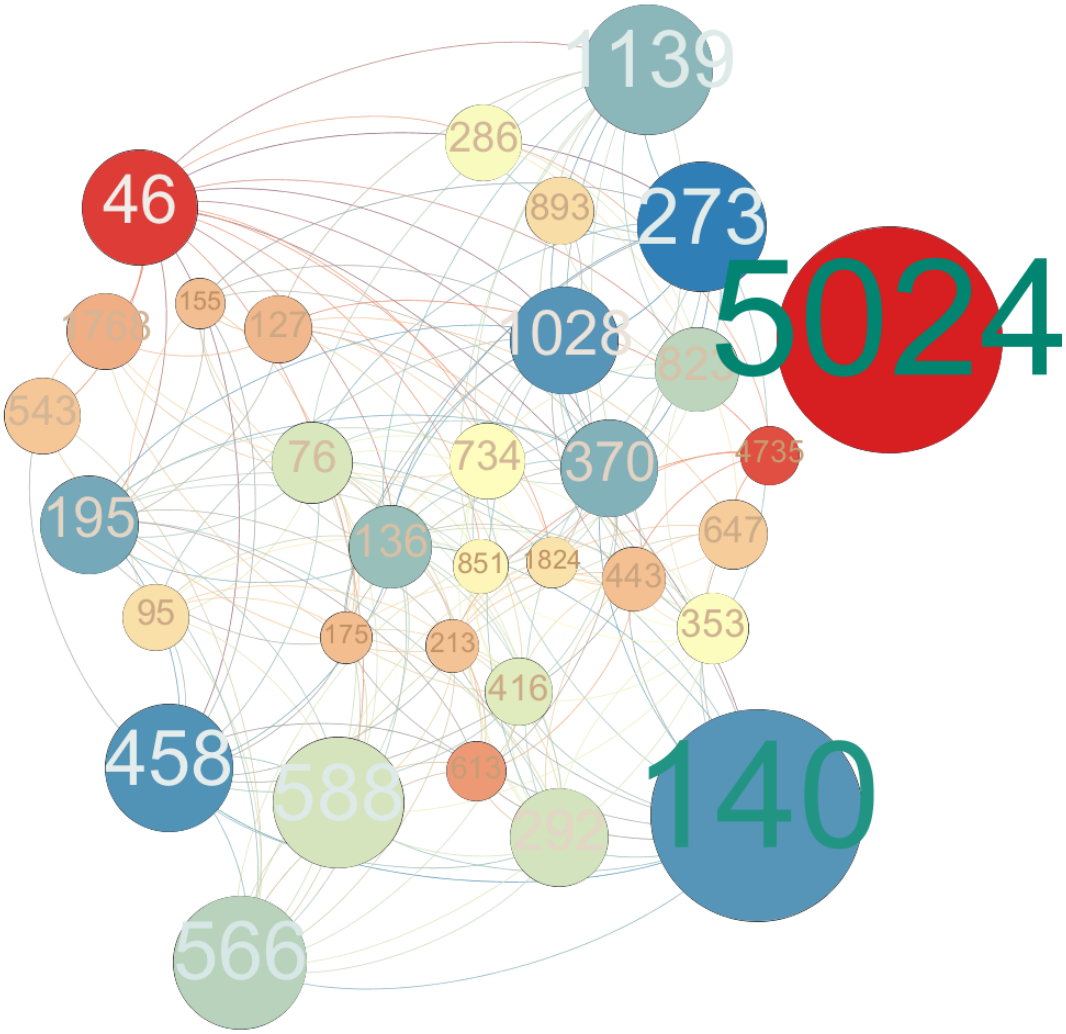

76
1139
5038
136
195
1028
416
273
353
370
458
588
734
140
292
823
286
566

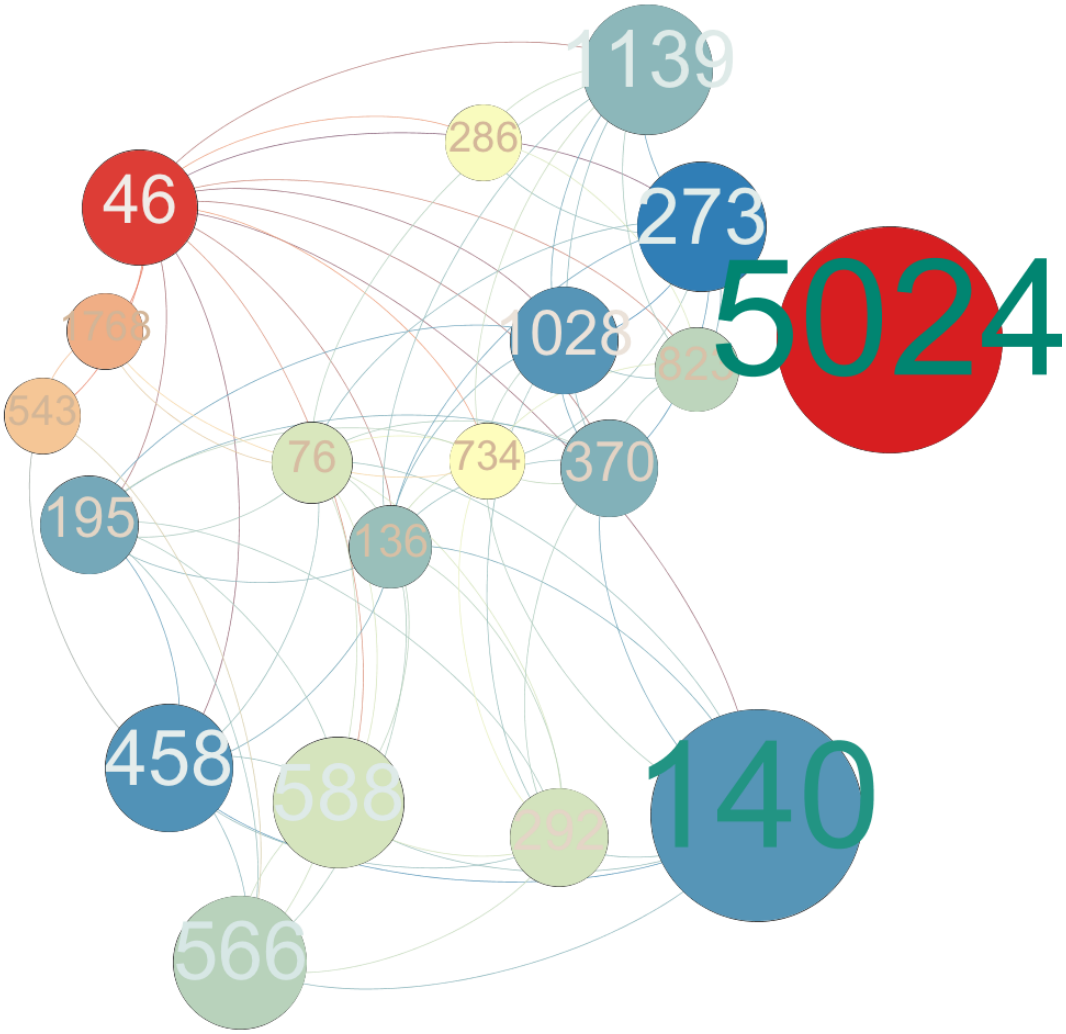


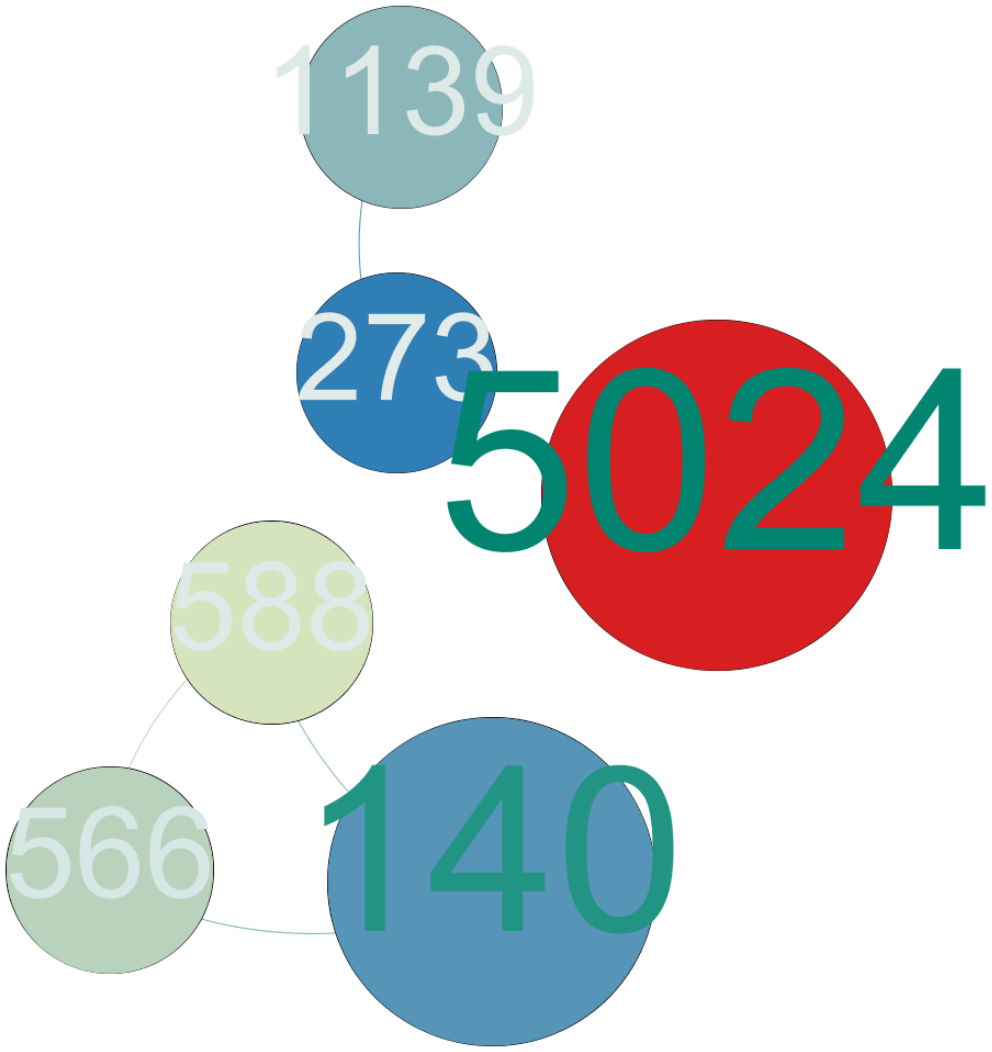
Enron betweenness centrality filters

```
In [18]: from IPython.display import Image
         from IPython.display import display

         aa=Image(filename='enbetcen50.png') #top50% to 1%
         bb=Image(filename='enbetcen10.png')
         cc=Image(filename='enbetcen05.png')
         dd=Image(filename='enbetcen01.png')

         display(aa,bb,cc,dd)
```


139
286
46
893
273
5024
1028
155
127
543
370
76
734
195
136
851
1824
647
443
353
95
175
516
213
478
530
416
458
140
588
292
566

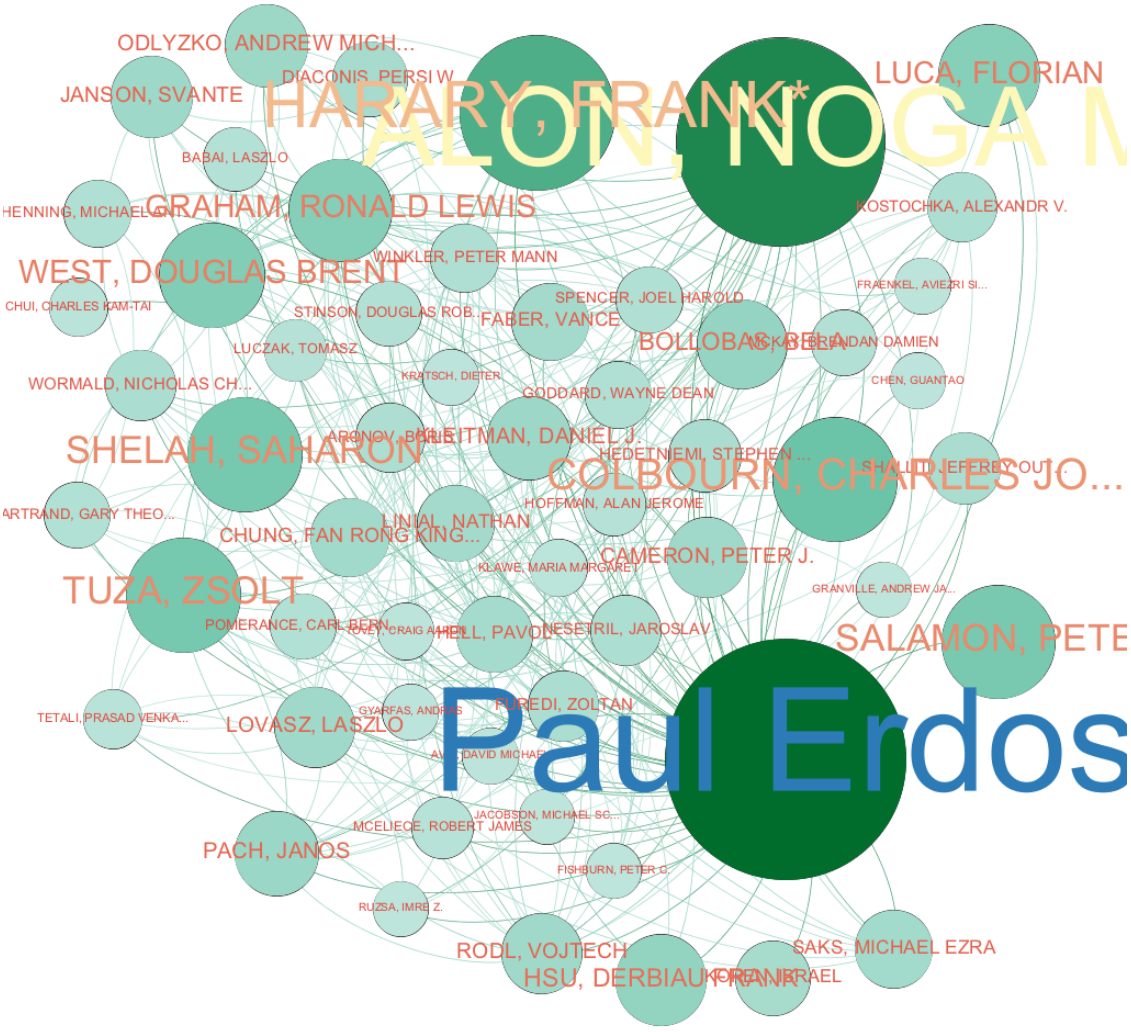

139
286
46
893
273
5024
155
768
127
1028
823
543
76
734
370
4735
195
136
647
851
1824
443
95
175
213
353
416
458
588
613
140
292
566

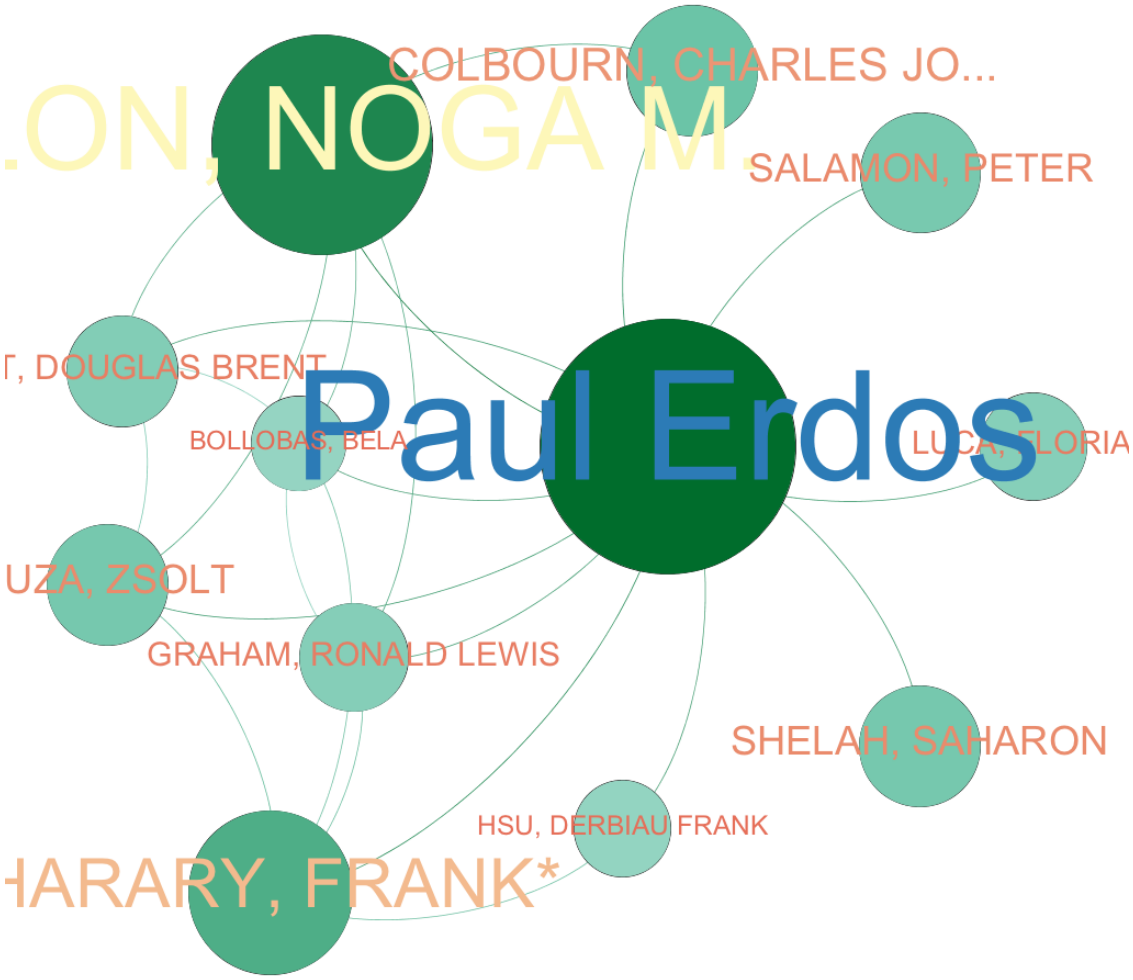

139
286
46
273
5024
1768
1028
823
543
76
734
370
195
136
458
588
140
292
566



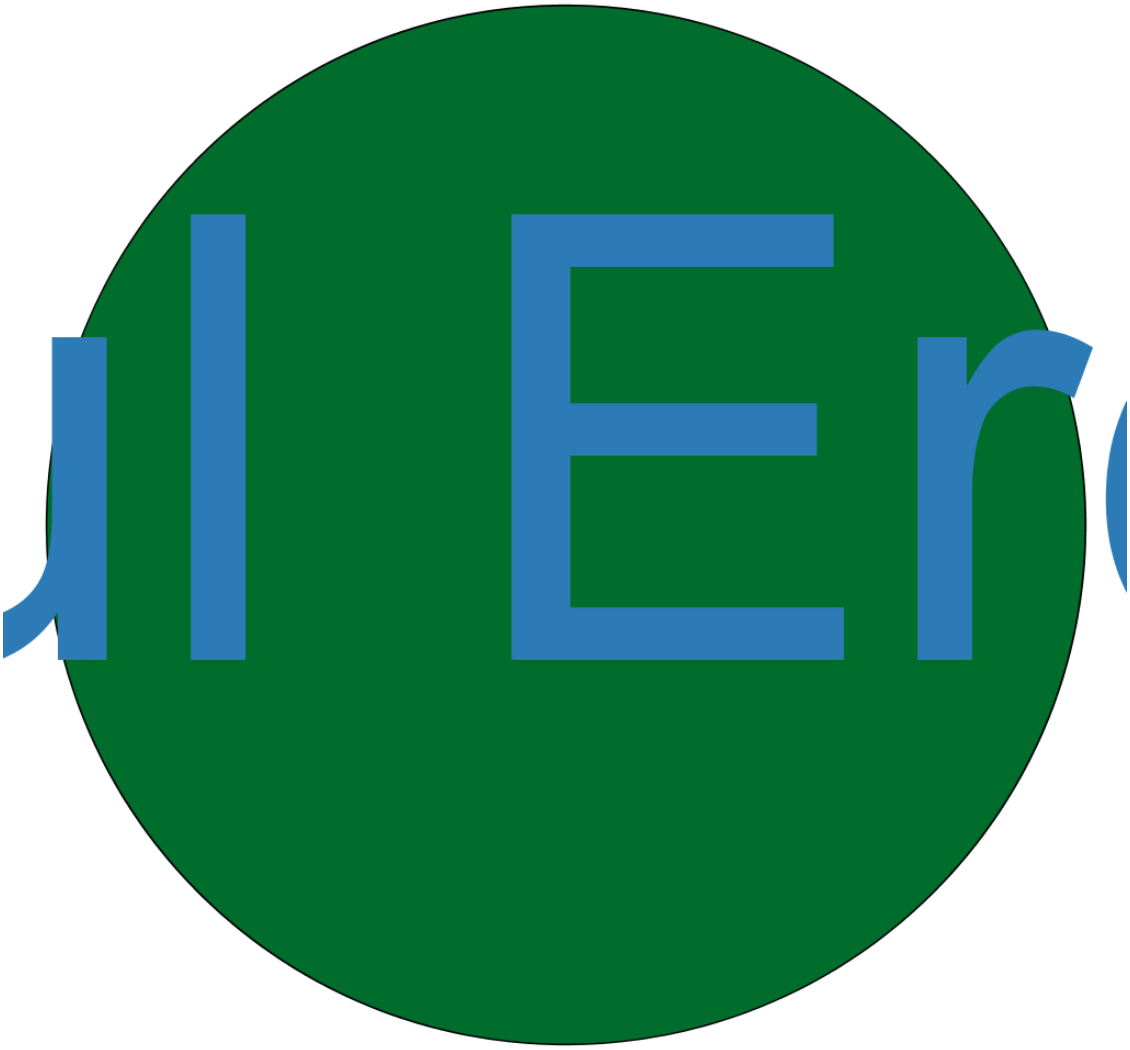
Erdos degree filters

```
In [21]: from IPython.display import Image
         from IPython.display import display

         e=Image(filename='erdosdeg50.png') #top50% to 1%
         f=Image(filename='erdosdeg10.png')
         g=Image(filename='erdosdeg05.png')
         h=Image(filename='erdosdeg01.png')

         display(e,f,g,h)
```


ODLYZKO, ANDREW MICH...
DIACONIS, PERSI W.
JANSON, SVANTE
HARARY, FRANK*
ALON, NO
GA M
LUCA, FLORIAN
BABAI, LASZLO
HENNING, MICHAEL A.
GRAHAM, RONALD LEWIS
ROSTOCHIKA, ALEXANDR V.
WINKLER, PETER MANN
WEST, DOUGLAS BRENT
FRAENKEL, AVIEZRI SI...
SPENCER, JOEL HAROLD
CHUI, CHARLES KAM-TAI
STINSON, DOUGLAS ROB...
FABER, VANCE
BOLLOBAS, BELA
ASKAR, BRENDAN DAMIEN
LUCZAK, TOMASZ
WORMALD, NICHOLAS CH...
KRATSCH, DIETER
CHEN, GUANTAO
GODDARD, WAYNE DEAN
SHELAH, SAHARON
ARONOV, BORIS
BEITMAN, DANIEL J.
HEDETNIEMI, STEPHEN ...
COLBOURN, CHARLES JO...
HOFFMAN, ALAN JEROME
ARTRAND, GARY THEO...
LINIAL, NATHAN
CHUNG, FAN RONG KING...
CAMERON, PETER J.
KLAWE, MARIA MARGARET
TUZA, ZSOLT
GRANVILLE, ANDREW JA...
POMERANCE, CARL BERN...
COLE, CRAIG A.
HELL, PAVOL
NESETRIL, JAROSLAV
SALAMON, PETE
Paul Erdos
PUREDI, ZOLTAN
TETALI, PRASAD VENKA...
LOVASZ, LASZLO
GYARFAS, ANDRAS
AV..., DAVID MICHAE...
JACOBSON, MICHAEL SC...
MCELIECE, ROBERT JAMES
PACH, JANOS
FISHBURN, PETER C.
RUZSA, IMRE Z.
RODL, VOJTECH
SAKS, MICHAEL EZRA
HSU, DERBIAU FRANK
KORAN, ISRAEL

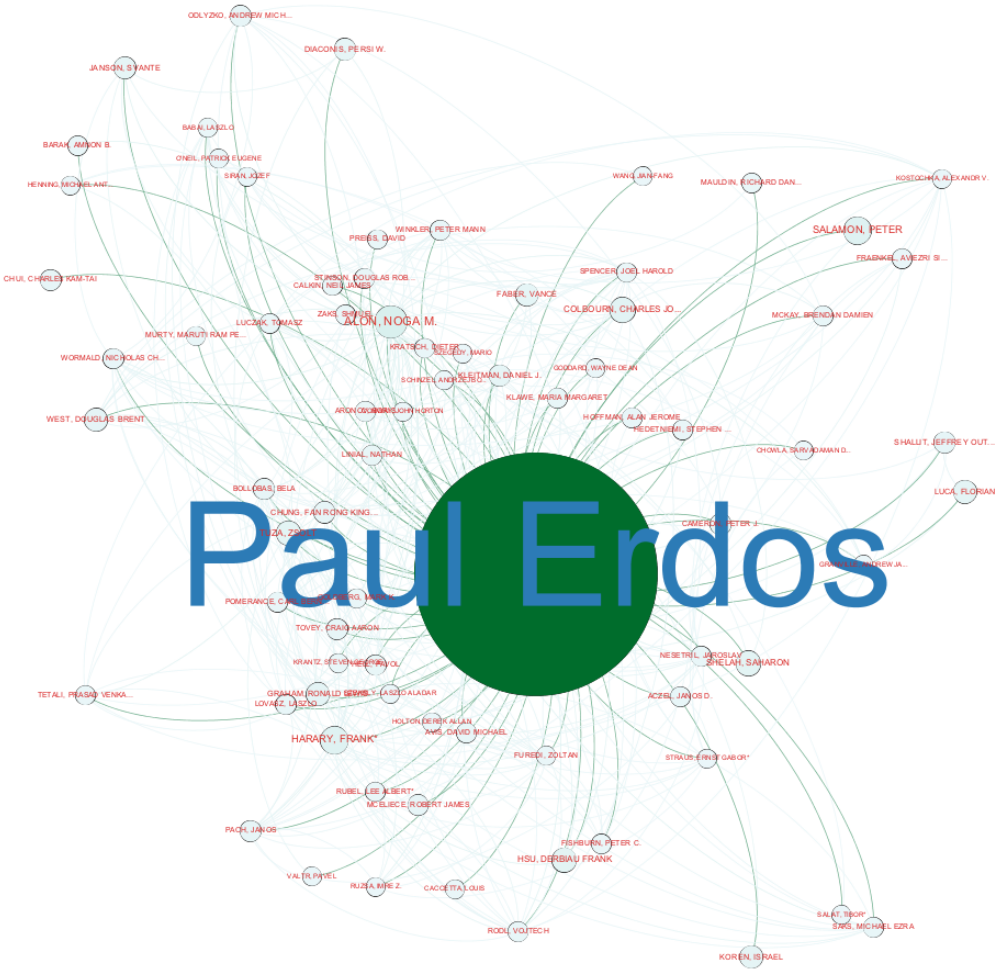

COLBOURN, CHARLES JO...
.ON, NOGA M.
SALAMON, PETER
Paul Erdos
BOLLOBAS, BELA
UZA, ZSOLT
GRAHAM, RONALD LEWIS
SHELAH, SAHARON
HSU, DERBIAU FRANK
HARARY, FRANK*

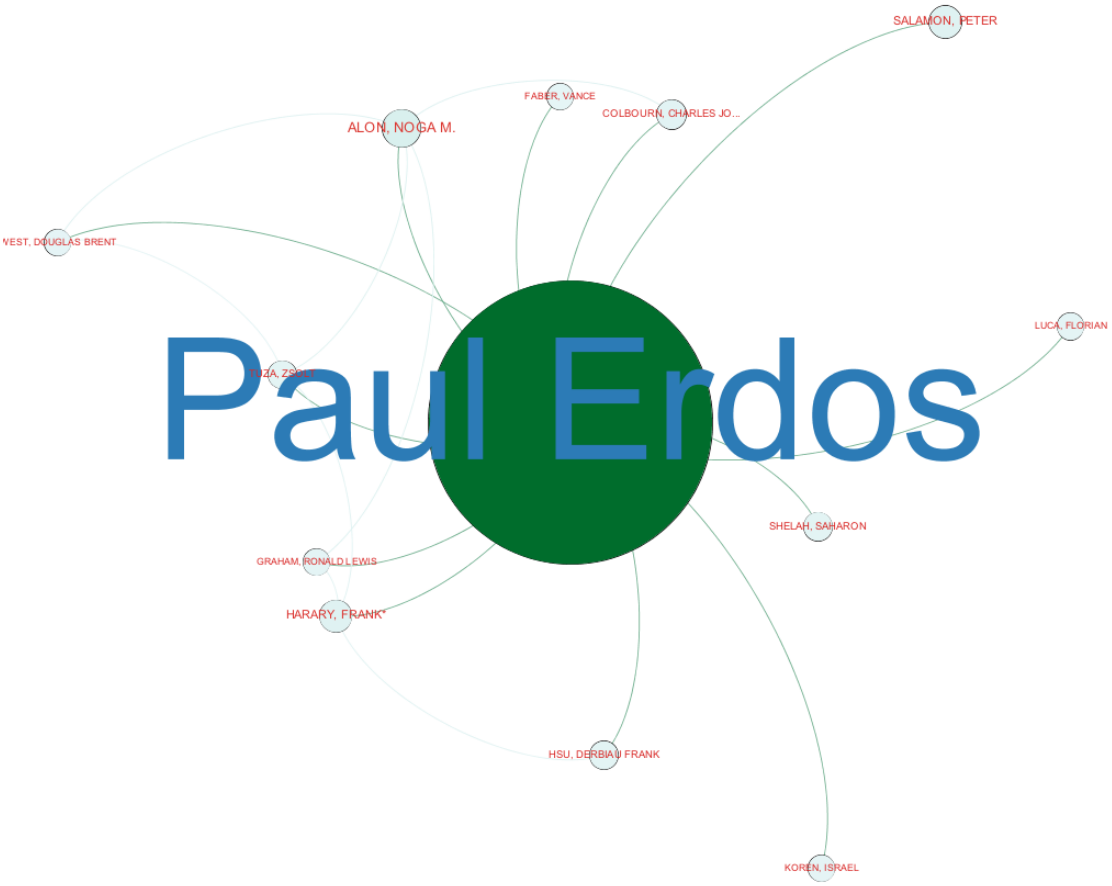

N, NOGAM.
COLBOURN, CHARLES J
UZA, ZSOLT
Paul Erdos
SHELAH, SAHA
ARY, FRANK*

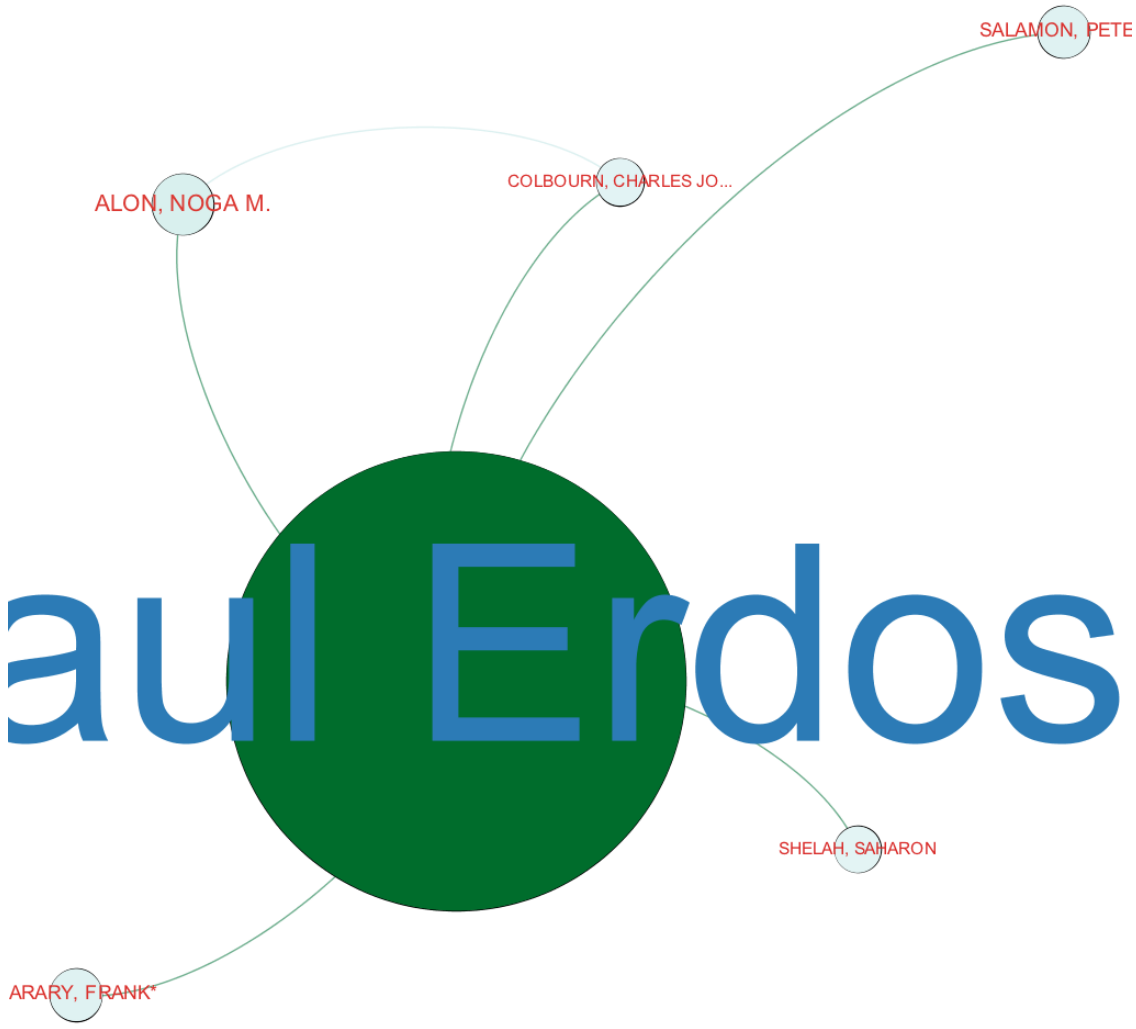


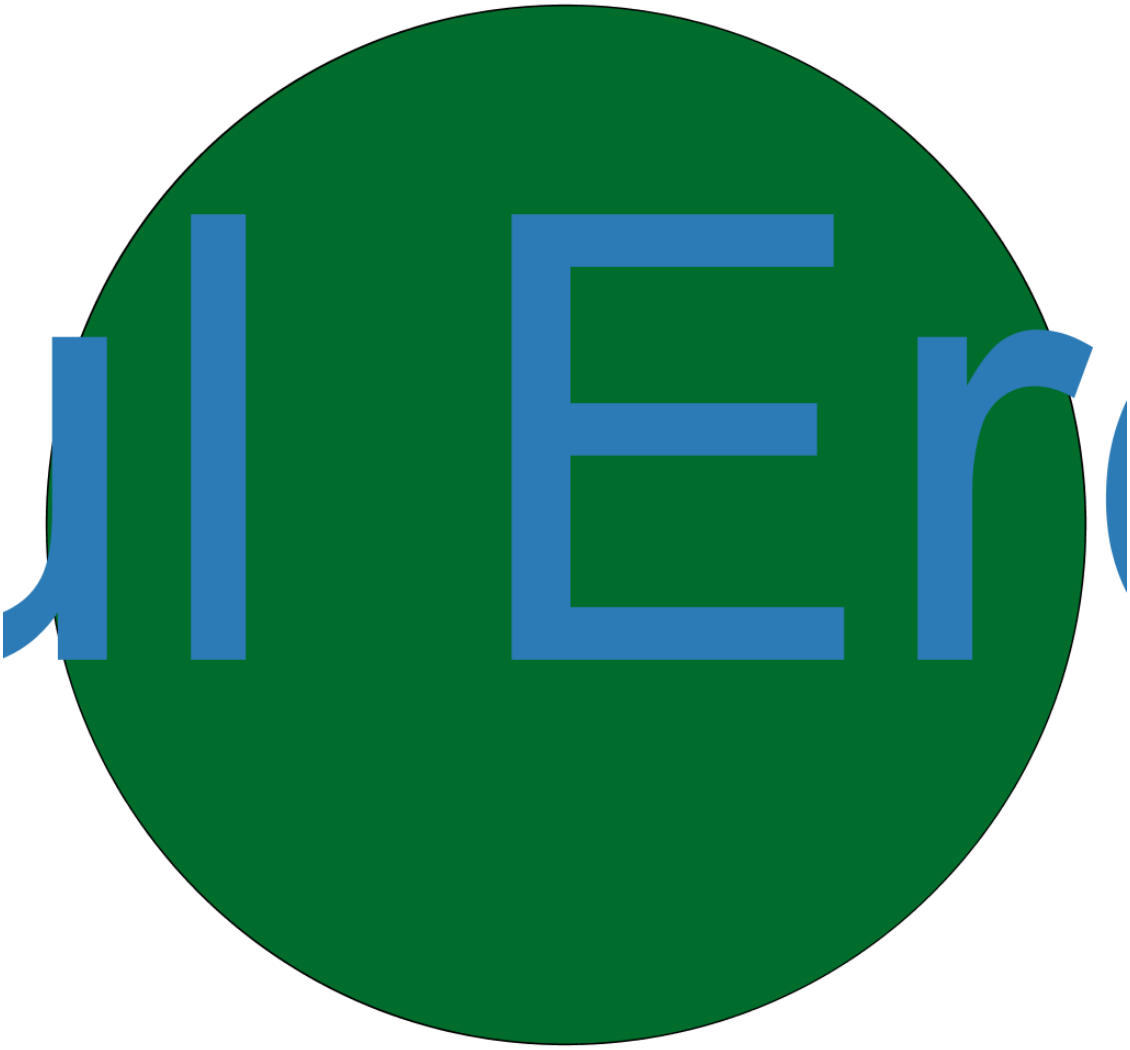
Erdos betweenness centrality filters

```
In [22]: from IPython.display import Image
         from IPython.display import display

         ee=Image(filename='erdosbetcen50.png') #top50% to 1%
         ff=Image(filename='erdosbetcen10.png')
         gg=Image(filename='erdosbetcen05.png')
         hh=Image(filename='erdosbetcen01.png')

         display(ee,ff,gg,hh)
```


Paul Erdos
ODLYZKO, ANDREW MICH...
DIACONIS, PERSI W.
JANSON, SVANTE
BABAI, LASZLO
BARAK, AMNON B.
O'NEIL, PATRICK EUGENE
SPAN, JOEF
HENNING, MICHAEL ANT...
WANG, JIANFANG
MAULDIN, RICHARD DAN...
KOSTOCHKA, ALEXANDR V.
SALAMON, PETER
FRAENKEL, AVIEZRI SI...
WINKLER, PETER MANN
PRIES, DAVID
SPENCER, JOEL HAROLD
CHUI, CHARLES KAM-TAI
STINSON, DOUGLAS ROB...
CALKIN, NEIL JAMES
FABER, VANCE
COLBOURN, CHARLES JO...
MCKAY, BRENDAN DAMIEN
ZAKS, SHMUEL
ALON, NOGA M.
LUCZAK, TOMASZ
MURTY, MARUTI RAM PE...
KRATSCH, DIETER
SZEKELY, MARIO
WORMALD, NICHOLAS CH...
GODDARD, WAYNE DEAN
SLEITMAN, DANIEL J.
SCHNITZER, ANDREAS
KLAWE, MARIA MARGARET
WEST, DOUGLAS BRENT
HOFFMAN, ALAN JEROME
HEDETNIEMI, STEPHEN ...
CHOMLA, SARVADAMAN D...
SHALLIT, JEFFREY OUT...
LINIAL, NATHAN
LUCA, FLORIAN
BOLLOBAS, BELA
CHUNG, FAN RONG KING...
CAMERON, PETER J.
POMERANCE, CARL
TOVEY, CRAIG AARON
NESETRIL, JAROSLAV
SHELAH, SAHARON
KRANTZ, STEVEN GEORGE
TETALI, PRASAD VENKA...
GRAHAM, RONALD LEWIS
LOVASZ, LASZLO
ACZEL, JANOS D.
HOLTON, DEREK ALAN
HARARY, FRANK*
AMS, DAVID MICHAEL
FUREDI, ZOLTAN
STRAUS, ERNST GABOR*
RUBEL, LEE ALBERT*
MCELIECE, ROBERT JAMES
PACH, JANOS
FISHBURN, PETER C.
HSU, DERBIAU FRANK
VALTR, PAVEL
RUZSA, IMRE Z.
CACCETTA, LOUIS
SAUER, NORBERT
STAS, MICHAEL EZRA
RODL, VOJTECH
KOREN, ISRAEL


SALAMON, PETER
FABER, VANCE
COLBOURN, CHARLES JO...
ALON, NOGA M.
'EST, DOUGLAS BRENT
LUCA, FLORIAN
Paul Erdos
SHELAH, SAHARON
GRAHAM, RONALD LEWIS
HARARY, FRANK*
HSU, DERBIA(I FRANK
KOREN, ISRAEL


SALAMON, PETE
ALON, NOGA M.
COLBOURN, CHARLES JO...
aul Erdos
SHELAH, SAHARON
ARARY, FRANK*



0.2.1 Part (B): Community Detection

In this task, we will try to find communities in the given netwok and explore more about them. NetworkX has built in functions for community detection (http://perso.crans.org/aynaud/communities/). Along with NetworkX, we will also use igraph library in this task, for community detection purposes.

```
In [19]: #install required packages and read their documentation to get used to them.
         !pip install community
         !pip install igraph
```

Community detection is a very common task for almost all networks. It helps us to understand network structure in much detail.

More information on community detection: https://arxiv.org/abs/0906.0612

0.2.2 There are multiple algorithms to detect communities. One of the commonly used algorithm is Louvain method. The method is a greedy optimization method that attempts to optimize the "modularity" of a partition of the network. 'community' library uses Louvain algorithm, and hence we get partitions based on optimized modularity. Implement a python code using 'community' library to find communities in the Citation network and Collaboration Network (Erdos). Write your code in the next cell and visualize your community detection results in Gephi for both the networks. Label the nodes in the visualization properly. Use largest connected components, if required. Include image(.jpg, .png) of the visualization here.

In [33]:
```
import community

#your code here
part33=community.best_partition(g33)

# size = float(len(set(part33.values())))
# pos = nx.spring_layout(g33)
# count = 0.
# for com in set(part33.values()) :
#       count += 1.
#       list_nodes = [nodes for nodes in part33.keys() if part33[nodes] == com]
#       nx.draw_networkx_nodes(g33, pos, list_nodes, node_size = 20,node_color = str(cou
# nx.draw_networkx_edges(g33,pos, alpha=0.5)
# plt.show()
```

In [34]:
```
# visualizations
#print len(part33)
communitydict33={}
c33=0
while(c33<len(dict_authors)):
    key1=list(part33)[c33]
    key2=dict_authors[key1]
    communitydict33[key2] = part33[key1]
    c33+=1
#print communitydict33
nx.set_node_attributes(g3, communitydict33,"com")
g33=nx.relabel_nodes(g3,labelmap)
#nx.write_gml(g33,"erdoscom.gml")
#print "erdos community gml created"
```

cit-net community

In [35]:
```
import community

part44=community.best_partition(g44)

#size = float(len(set(part44.values())))
```

```
        #pos = nx.spring_layout(g44)
        #count = 0.
        #for com in set(part44.values()) :
        #     count += 1.
        #     list_nodes = [nodes for nodes in part44.keys() if part44[nodes] == com]
        #     nx.draw_networkx_nodes(g44, pos, list_nodes, node_size = 20,node_color = str(coun
        #nx.draw_networkx_edges(g44,pos, alpha=0.5)
        #plt.show()
```

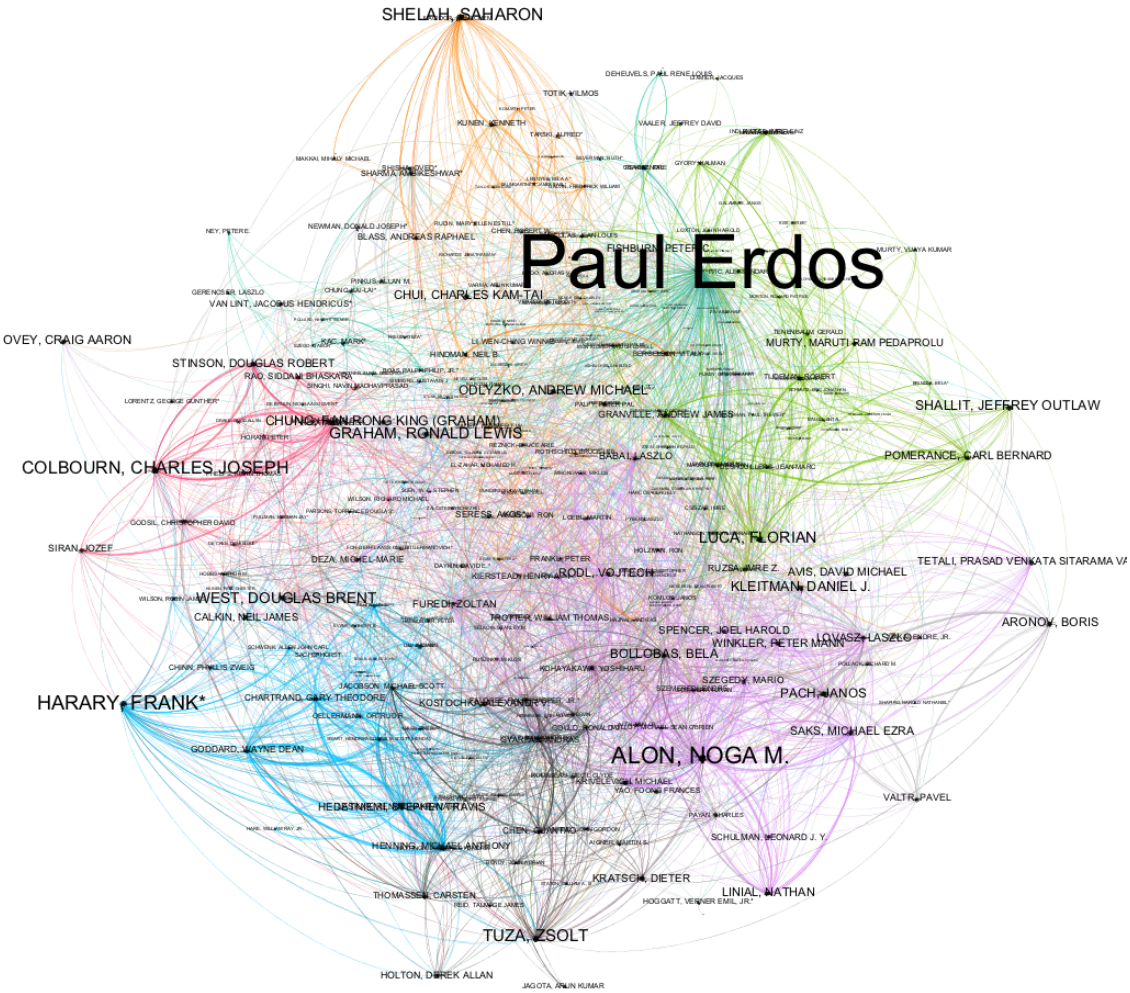
In [ ]:
```
%%time
from multiprocessing import Pool
import time
import itertools

def chunks(l, n):
    """Divide a list of nodes `l` in `n` chunks"""
    l_c = iter(l)
    while 1:
        x = tuple(itertools.islice(l_c, n))
        if not x:
            return
        yield x


def _betmap(G_normalized_weight_sources_tuple):
    """Pool for multiprocess only accepts functions with one argument.
    This function uses a tuple as its only argument. We use a named tuple for
    python 3 compatibility, and then unpack it when we send it to
    `betweenness_centrality_source`
    """
    return nx.betweenness_centrality_subset(*G_normalized_weight_sources_tuple)


def betweenness_centrality_parallel(G, processes=4):
    """Parallel betweenness centrality  function"""
    p = Pool(processes=processes)
    node_divisor = len(p._pool)*4
    node_chunks = list(chunks(G.nodes(), int(G.order()/node_divisor)))
    #print node_chunks
    num_chunks = len(node_chunks)


    bt_sc = p.map(_betmap,
                  zip([G]*num_chunks,
                      node_chunks,
                      [list(G)]*num_chunks,
                      [True]*num_chunks,
                      [None]*num_chunks))
```

```
            #print bt_sc

            # Reduce the partial solutions
            bt_c = bt_sc[0]
            for bt in bt_sc[1:]:
                for n in bt:
                    bt_c[n] += bt[n]
            return bt_c


        print("")
        print("Computing betweenness centrality for:")
        print(nx.info(g44))
        print("\tParallel version")
        start = time.time()
        btcitnet = betweenness_centrality_parallel(g44)
        print("\t\tTime: %.4F" % (time.time()-start))
        print("\t\tBetweenness centrality for node 0: %.5f" % (btcitnet[0]))
        print("")

        %store btcitnet > citnetbetCen.txt


Computing betweenness centrality for:
Name:
Type: Graph
Number of nodes: 27400
Number of edges: 352059
Average degree:  25.6977
        Parallel version


In [ ]: import os
        #%store btcitnet > citnetbetCen.txt
        dc=nx.degree_centrality(g44)
        print "1"
        for k in g44.nodes():
            g44.node[k]['dc'] = dc[k]

        print "2"
        def get_authors(l):
            authors = reduce(list.__add__, [a.split(",") for a in l[9:].split("and")])
            return [x.strip() for x in authors]

        cit_auth2={}
        cit_auth={}
```

```
        thred = sorted(dc.values(), reverse=True)[100]
        for subdir, dirs, files in os.walk("cit-HepTh-abstracts"):
            for fl in files:
                filepath = subdir + os.sep + fl

                if filepath.endswith(".abs"):
                    node_num = int(fl[:-4])
                    name = ""
                    for l in open(filepath):
                        if l.startswith("Authors:"):
                            name = get_authors(l)[0]
                    if node_num in g44.nodes():
                        if g44.node[node_num]['dc'] > thred:
                            g44.node[node_num]['author'] = name
                            cit_auth2[node_num]=name
                        else:
                            cit_auth[node_num]=node_num

        cit=0
        while(cit<len(cit_auth2)):
            key1=list(cit_auth2)[cit]
            cit_auth[key1]=cit_auth2[key1]
            cit+=1

        #print cit_auth
        nx.set_node_attributes(g44, btcitnet,"betCen")
        nx.set_node_attributes(g44, part44,"com")
        #nx.set_node_attributes(g44, cit_auth,"author")
        g44=nx.relabel_nodes(g44,cit_auth)
        nx.write_gml(g44,"citnetcom.gml")


        #needed to make png with authornames instead of node ids
        #nx.write_gml(g44,"citnetcom.gml")
```

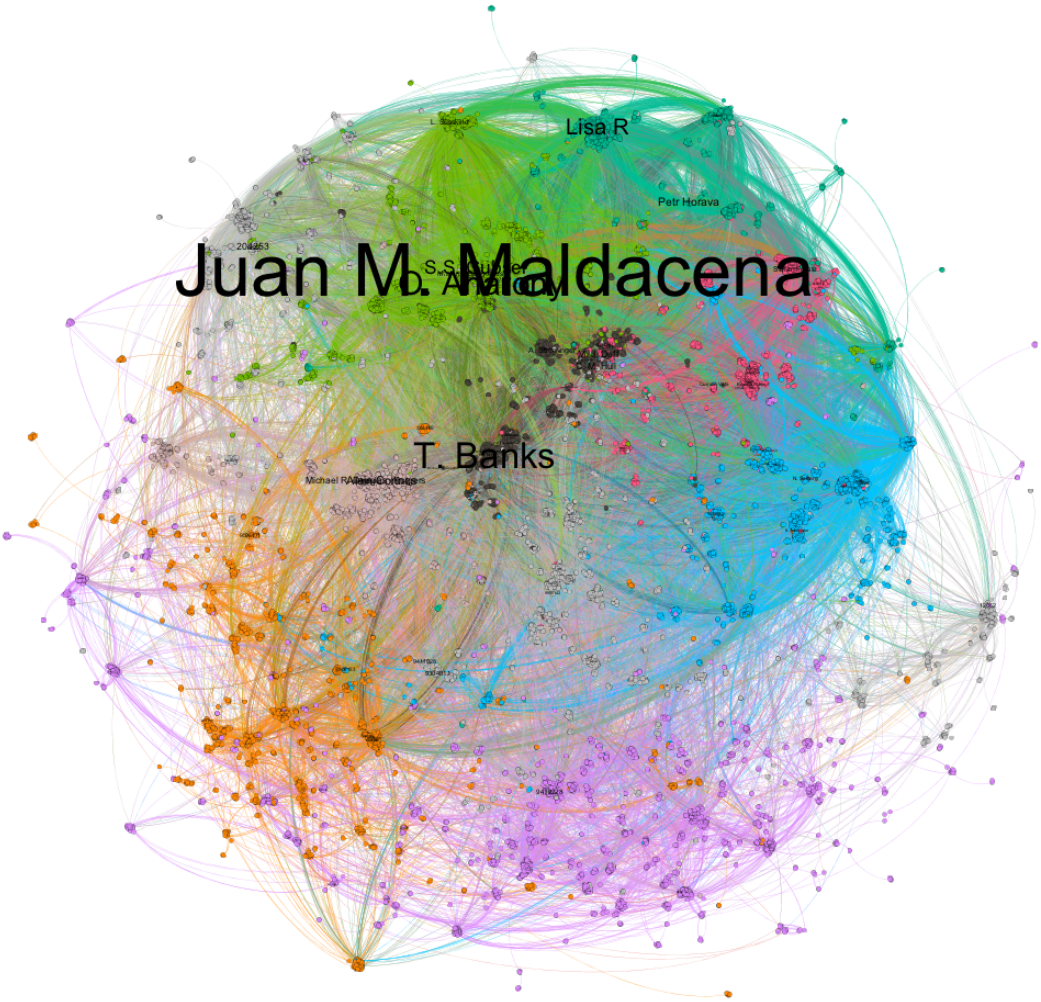
Erdos Community using Louvain

```
In [61]: from IPython.display import Image
         from IPython.display import display

         commL =Image(filename='erdoscomt3.png') #top50% to 1%
         #commG =Image(filename='erdosgreedycomt1.png')
         #commEV =Image(filename='erdoscomevt1.png')

         display(commL)
```



citnet Community using Louvain

```
In [64]: from IPython.display import Image
         from IPython.display import display

         commL1 =Image(filename='citnetcomt1.png') #top50% to 1%
         #commG =Image(filename='erdosgreedycomt1.png')
         #commEV =Image(filename='erdoscomevt1.png')

         display(commL1)
```


Lisa R
Petr Horava
Juan M. Maldacena
T. Banks

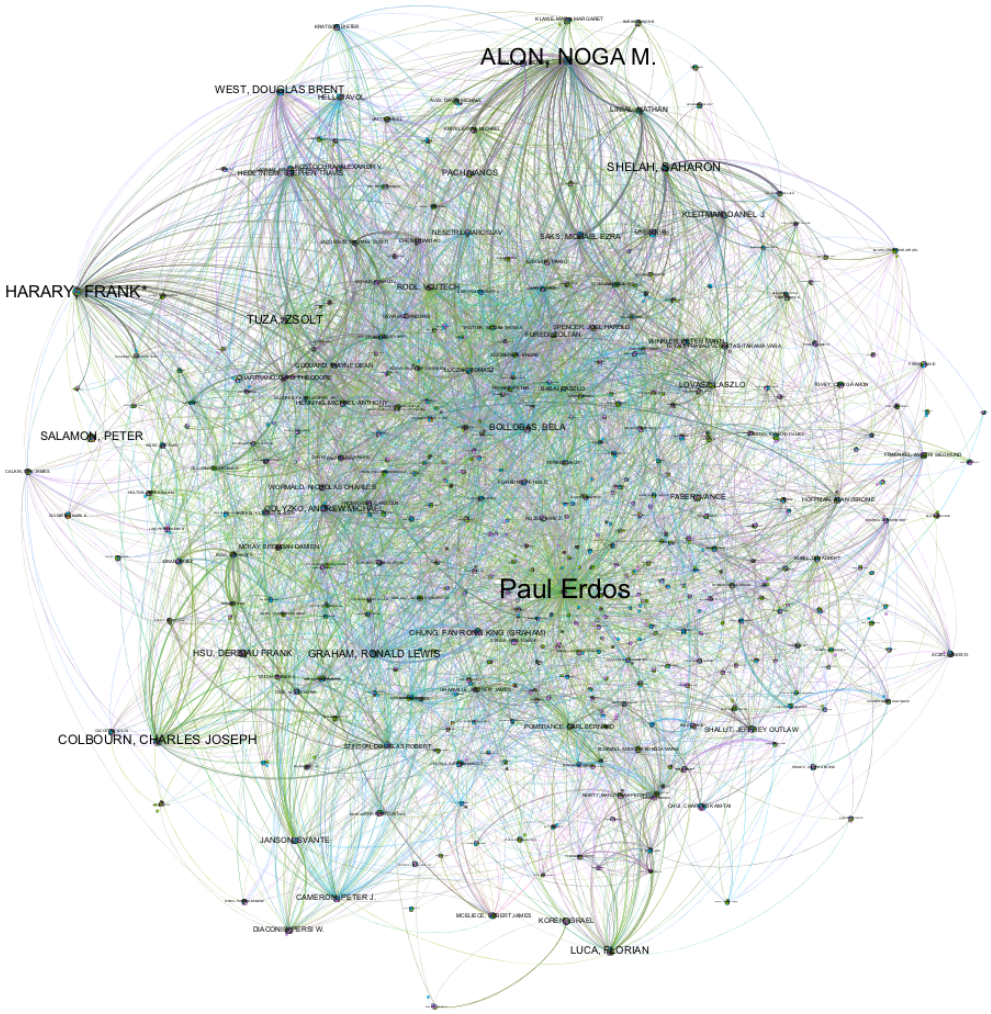
0.2.3 Compared to 'community' library, 'igraph' has more flexibilty to detect communities. igraph allows user to partition the network in the number of communities that user wishes. Obviously this number is bounded. Now, you will use this aspect to divide given network in '5' communities using 'igraph' and observe the results. Also, derive results for optimized modularity condition in igraph. Write a python code to implement above task for citation network & collaboration network (Erdos). Remember that unlike 'community', igraph has multiple approach for community detection. Obvious approach being greedy and it optimizes modularity. Visualize your community detection results in Gephi for both the networks. Label the nodes in the visualization properly. Use largest connected components, if required. Use different colors for nodes in every community. Include image(.jpg, .png) of the visualization here.

In [55]:
```
import igraph
#from igraph import *

#partition network using greedy approach. Note the number of communities

#citnetwork
newG1=igraph.read("citnetcom.gml")
summary(newG1)
g1com = newG1.community_fastgreedy()
g1comlist=g1com.as_clustering().membership
newG1.vs["com_greedy"]=g1comlist
igraph.write(newG1,"citnetgreedy.gml")

#erdos network
newG2=igraph.read("erdoscom.gml")
summary(newG2)
g2com = newG2.community_fastgreedy()
g2comlist=g2com.as_clustering().membership
newG2.vs["com_greedy"]=g1comlist
igraph.write(newG2,"erdosgreedy.gml")

#partition network in 5 communities and see the difference in the visualization
g15comlist=g1com.as_clustering(5).membership
newG1.vs["com_greedy5"]=g1comlist
igraph.write(newG1,"citnetgreedy5.gml")


g25comlist=g2com.as_clustering(5).membership
newG2.vs["com_greedy5"]=g1comlist
igraph.write(newG2,"erdosgreedy5.gml")

#print len(g1comlist)
#print len(g2comlist)
#print len(g15comlist)
#print len(g25comlist)
```

```
IGRAPH U--- 27359 342819 --
+ attr: author (v), betCen (v), com (v), dc (v), id (v), label (v)
IGRAPH U--- 11524 18504 --
+ attr: author (v), betCen (v), com (v), deg (v), id (v), label (v)
```

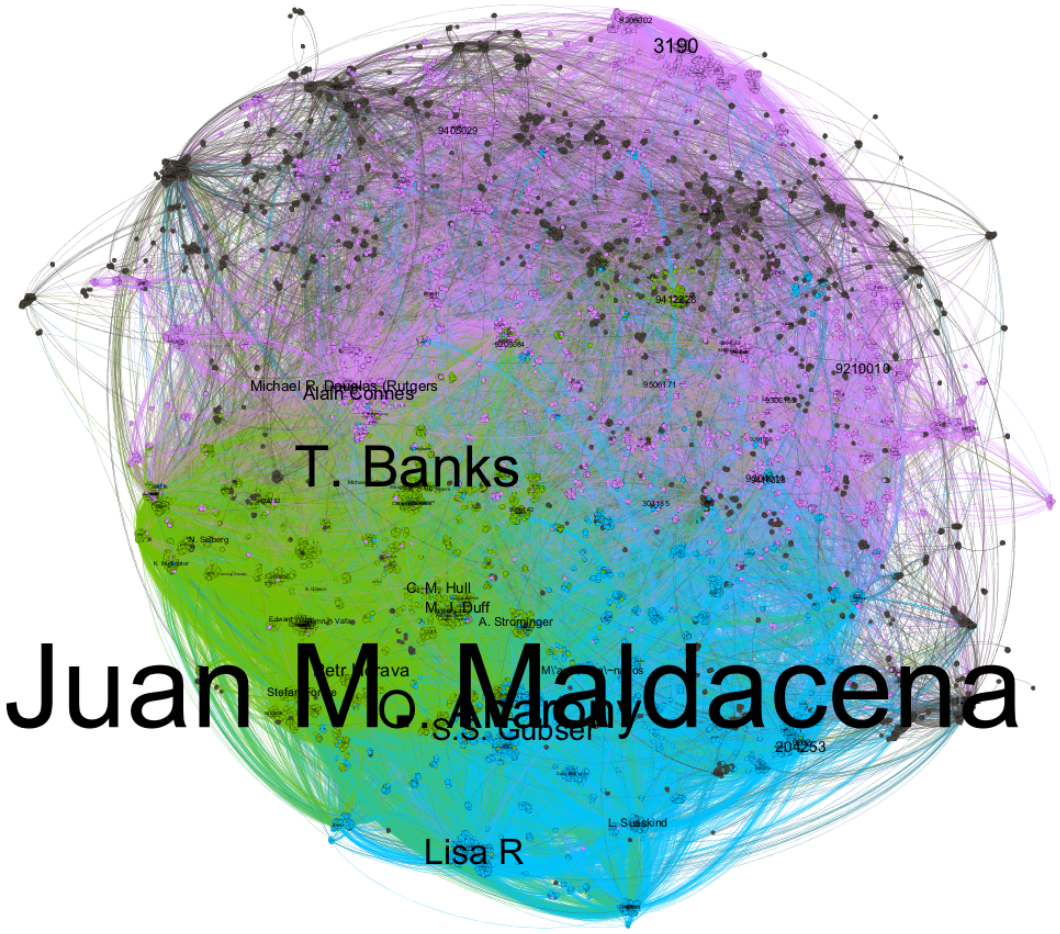
Erdos greedy comm detection

In [62]:
```
from IPython.display import Image
from IPython.display import display

#commL =Image(filename='erdoscomt3.png') #top50% to 1%
commG =Image(filename='erdosgreedycomt1.png')
#commEV =Image(filename='erdoscomevt1.png')

display(commG)
```



Citnet greedy comm detection

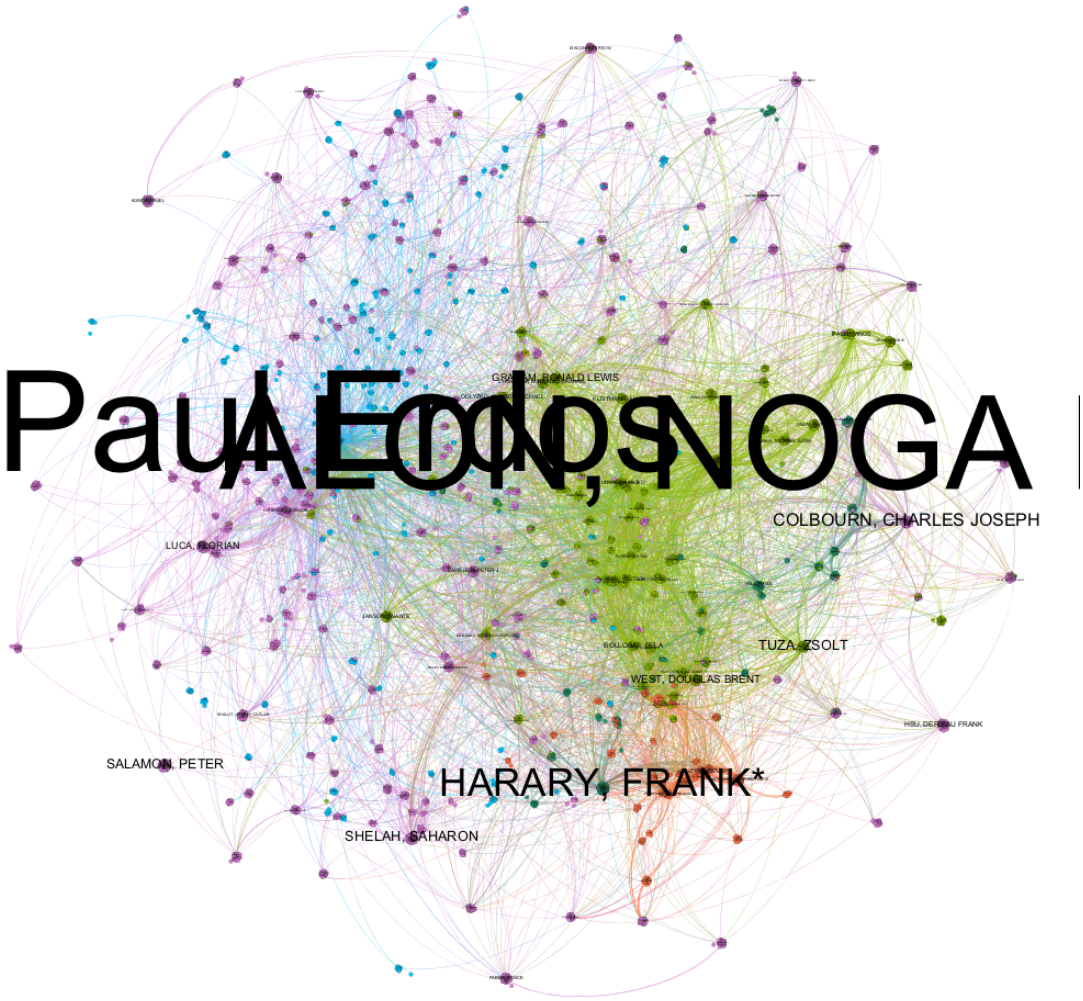
```
In [65]: from IPython.display import Image
         from IPython.display import display


         commG1 =Image(filename='citnetgreedycomt2.png')

         display(commG1)
```



Erdos 5 Comm detection

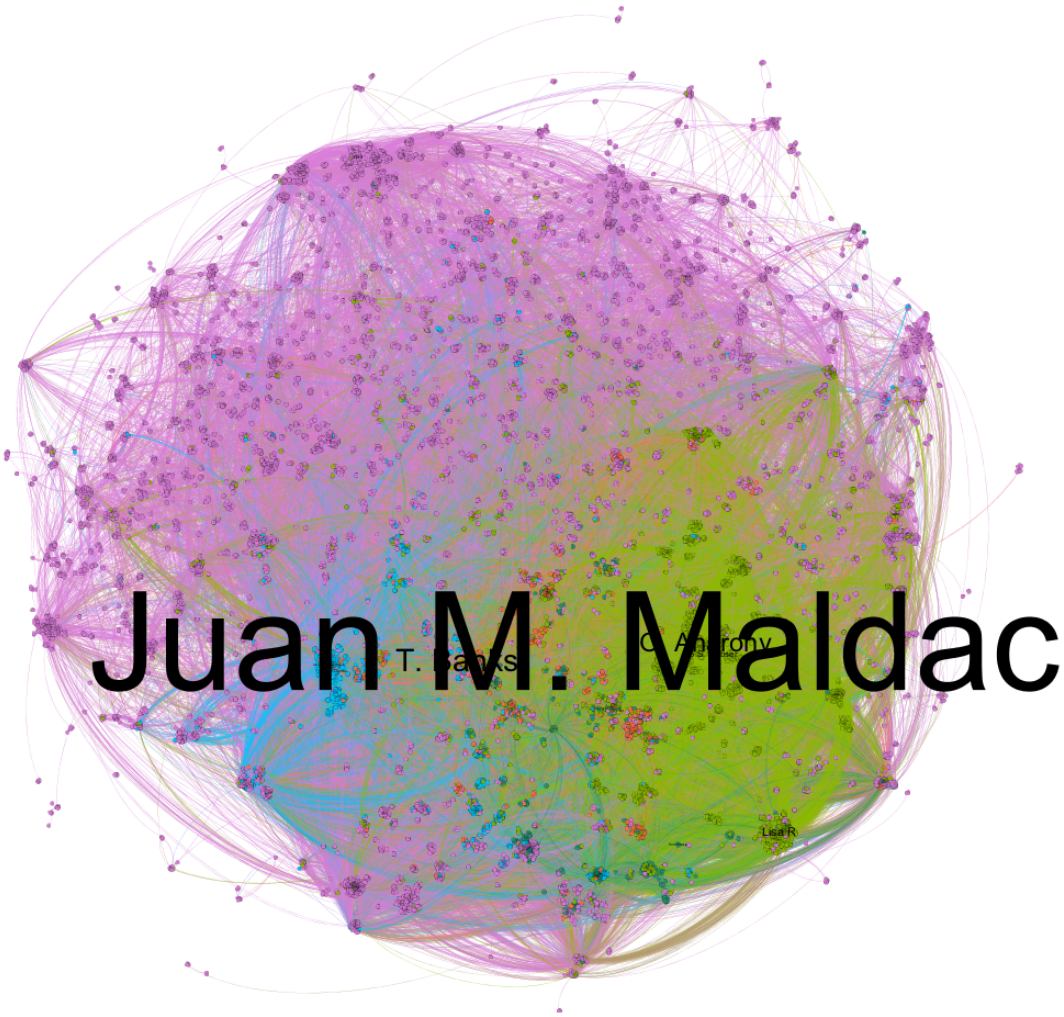
In [57]:
```
#visualizations

g11com=newG1.community_leading_eigenvector(5)
g11comlist=g11com.membership
newG1.vs["com_ev"]=g11comlist
igraph.write(newG1,"citnetev5.gml")


g22com=newG2.community_leading_eigenvector(5)
g22comlist=g22com.membership
newG2.vs["com_ev"]=g22comlist
igraph.write(newG2,"erdosev5.gml")
```

In [63]:
```
from IPython.display import Image
from IPython.display import display

commEV =Image(filename='erdoscomevt1.png')

display(commEV)
```



Citnet 5 comm detection

```
In [66]: from IPython.display import Image
         from IPython.display import display

         commEV1 =Image(filename='citnetcomevt1.png')

         display(commEV1)
```



0.2.4 Now that we have detected communities, we will further analyze our results. This task is only for Collaboration network (Erdos). Use results from communtiy detection using 'community'. Sort communities and get largest 5 communities. For each of these 5 communities, get 3 nodes with the highest node degree. So you will get 3 authors per community, for 5 communities. Now search the area of research for each of these authors and enlist them. Further, observe if there is any reason for those 3 authors to be in same community, for each community. State that reason in brief. Write all of your results in next cell. Also include any other interesting results that you may observe during process.

1 your observations here

community 10
(1)Alon, Noga

(2)Lovasz, Laszlo
(3)Rodl,Vojtech
Mathmeticians that research combinatorics and other areas in graph theory
community 14
(1)Luca, Florian
(2)Shallit, Jeffrey
(3)Pomerance, Carl
Research area focusing on number theory
community 1
(1)Harrary, Frank
(2)Hedetniemi, Stephen
(3)Henning, Michael
research area focusing on graph theory
community 22
(1)Tuza, Zsult
(2)Bollobas, Bela
(3)Gyarfas, Andras
All Hungarian mathematicians that have research in graph theory
community 6
(1)Shellah, Saharon
(2)Blass, Andreas
(3)Hindman, Neil
Their research focuses on set theory and mathematical logic.