Applications of Navigability and the Structure of Social Networks

Today

1)Related research paper and review of Geographic Routing

2)Measures of Centrality

Small World Effect

"Any individual in the world can reach any other individual through a short chain of social ties (*1*, *2*). Early experimental work by Travers and Milgram (*3*) suggested that the average

length of such chains is roughly six, and recent theoretical (*4*) and empirical (*4*–*9*)

work has generalized the claim to a wide range of nonsocial networks.

In particular, individuals in real social networks have only limited, local information

about the global social network and, therefore, finding short paths represents a nontrivial search effort (*10*–*12*)."

Source: **An Experimental Study of Search in Global Social Networks, Doods et al 2003 (attached)**

SIX DEGREES | 1967: Stanley Milgram

F.

,.

s.

o.

Modern version by email

Targets included a professor at an Ivy League university, an archival inspector in Estonia, a technology consultant in India, a policeman in Australia, and a veterinarian in the Norwegian army.

Participants were informed that their task was to help relay a message to their allocated target by passing the message to a social acquaintance whom they considered "closer" than themselves to the target. Of the 98,847 individuals who registered, about 25% provided their personal information and initiated message chains.

Modern Milgram's experiment to collectively help that a message to reach a random target person just by sending the message to one acquaintance.
Known of target: **Name, Profession and City of the target.**

Reasons for choosing next recipient. All quantities are percentages.

1967, Milgram Experiment 160 Random Participant From Omaha to Boston

L	N	Location	Travel	Family	Work	Education
1	19,718	33	16	11	16	3
2	7,414	40	11	11	19	4
3	2,834	37	8	10	26	6
4	1,014	33	6	7	31	8
5	349	27	3	6	38	12
6	117	21	3	5	42	15
7	37	16	3	3	46	19

Peter Sheridan Dodds, Roby Muhamad, and Duncan J Watts. **An experimental study of search in global social networks**. **Science**, 301(5634):827-829, 2003.

384 out of 24,164 Reached the target!

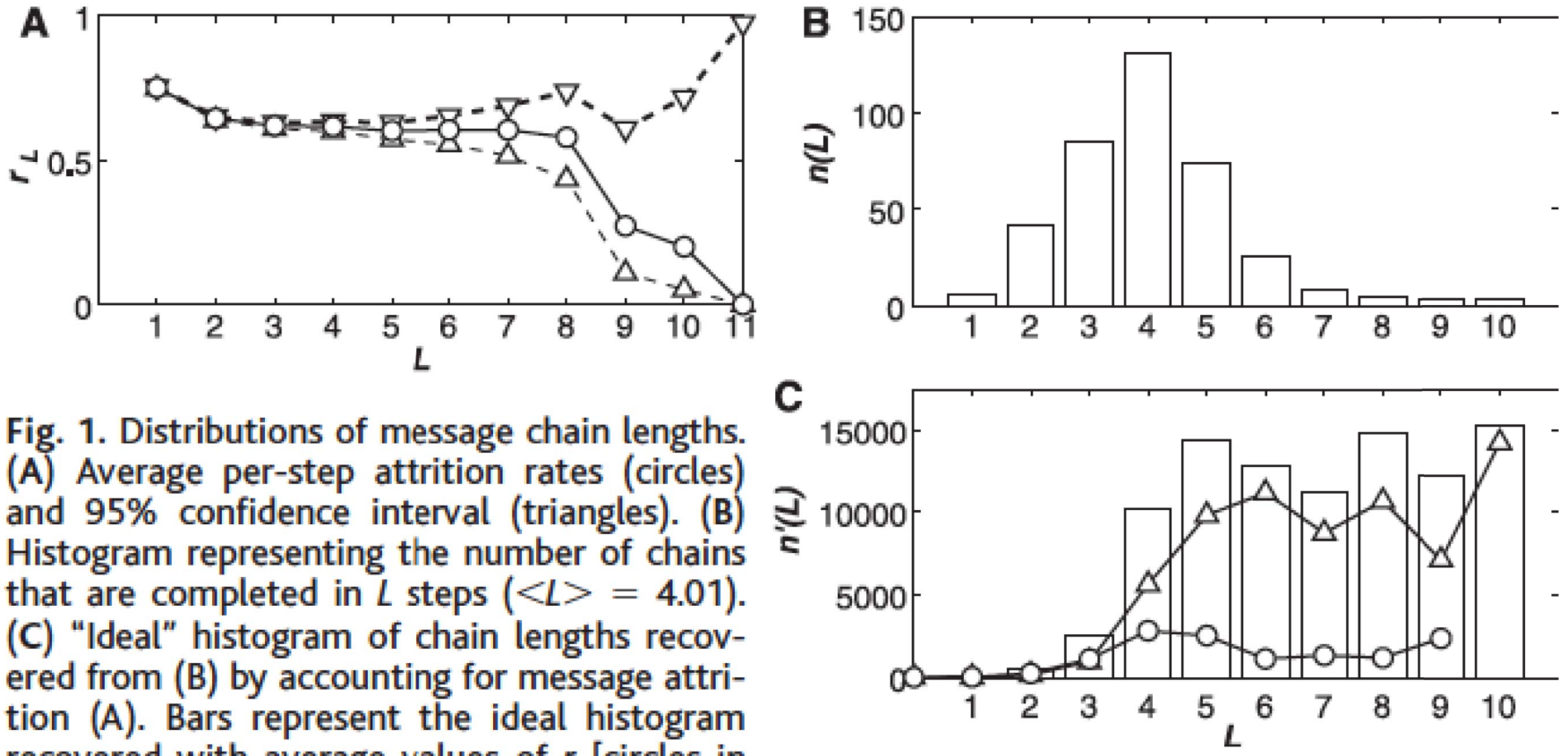
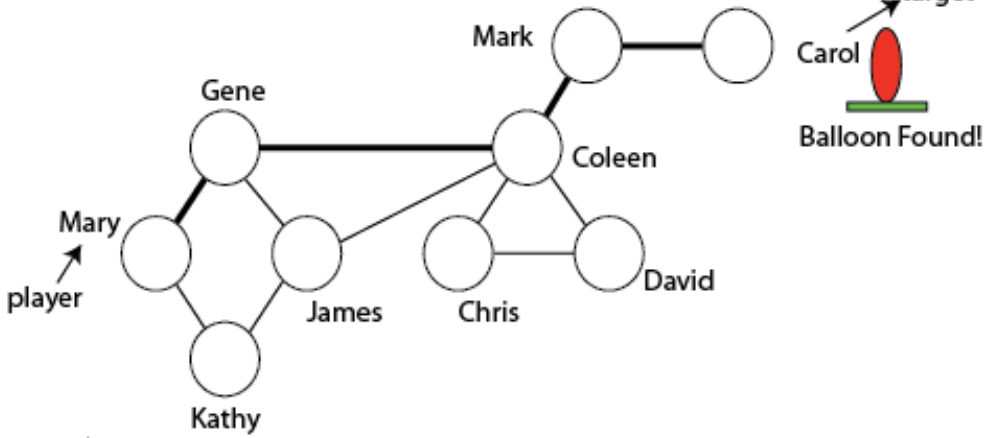


Fig. 1. Distributions of message chain lengths. (A) Average per-step attrition rates (circles) and 95% confidence interval (triangles). (B) Histogram representing the number of chains that are completed in L steps ($\langle L \rangle = 4.01$). (C) "Ideal" histogram of chain lengths recovered from (B) by accounting for message attrition (A). Bars represent the ideal histogram recovered with average values of r [circles in (A)] for the histogram in (B); lines represent a decomposition of the complete data into chains that start in the same country as the target (circles) and those that start in a different country (triangles).

Power of Social Network: Crowdsourcing



Carol wins $2,000
Mark wins $1,000
Coleen wins $500
Gene wins $250

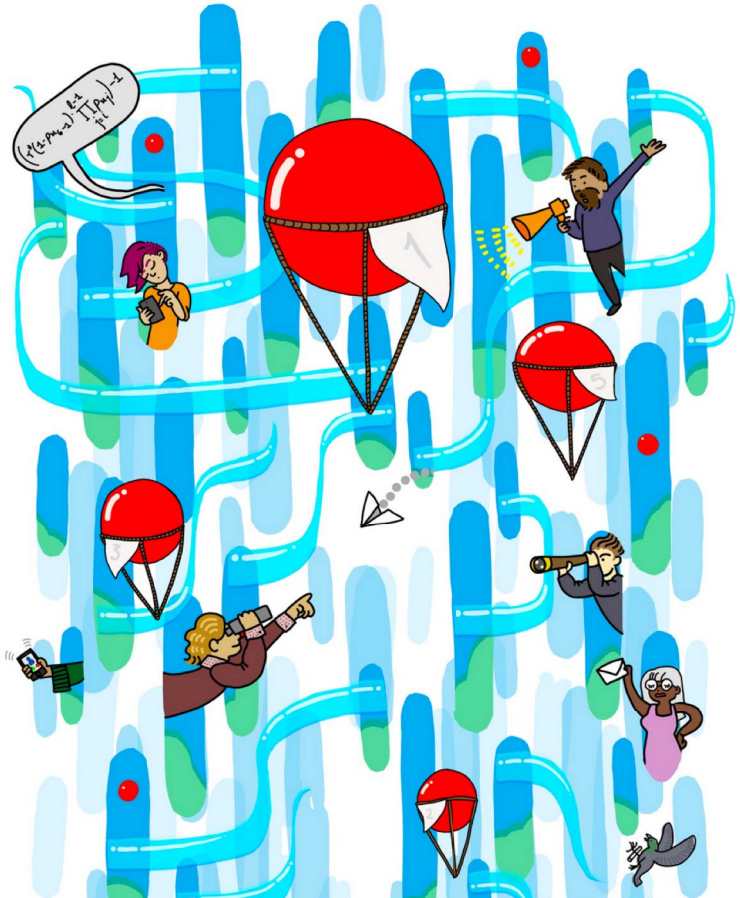
Red Balloon Experiment: 8 balloons randomly placed in the U.S. were found within 8 hours via social media

Manuel Cebrian Follow
Research Scientist
Feb 18, 2016 · 18 min read

SEARCHING FOR SOMEONE

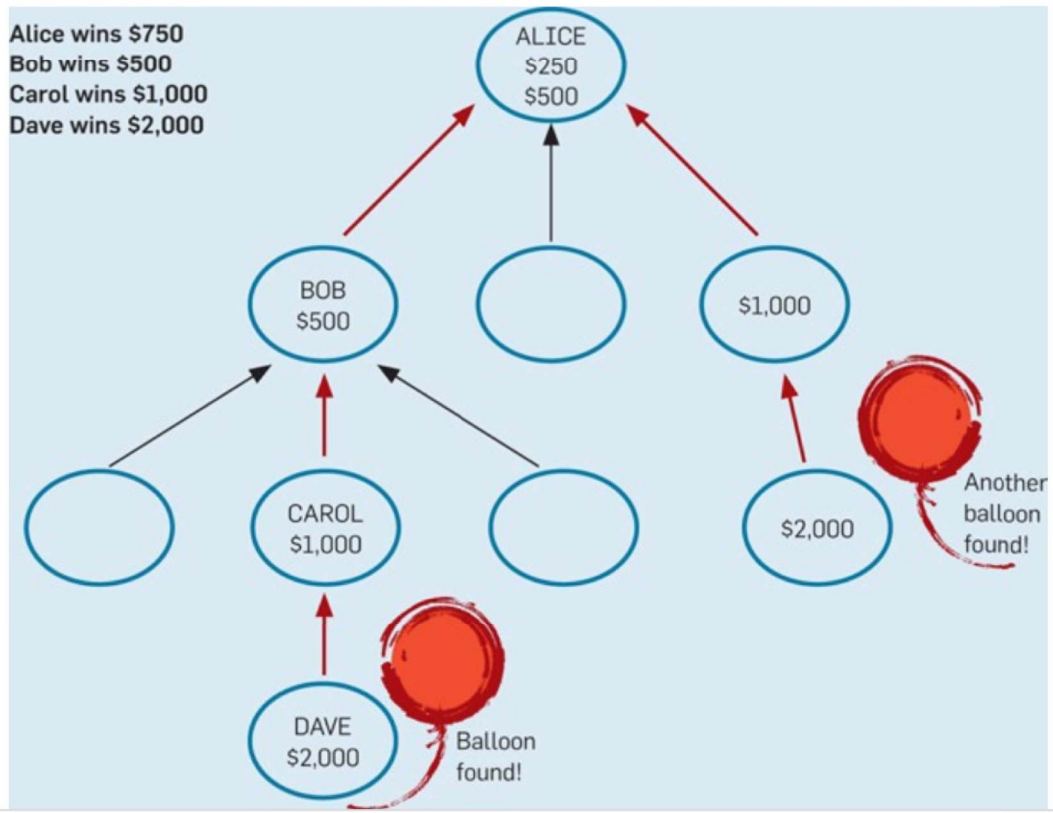
From the "Small World Experiment" to the "Red Balloon Challenge," and beyond

Crowdsourcing Annus Mirabilis

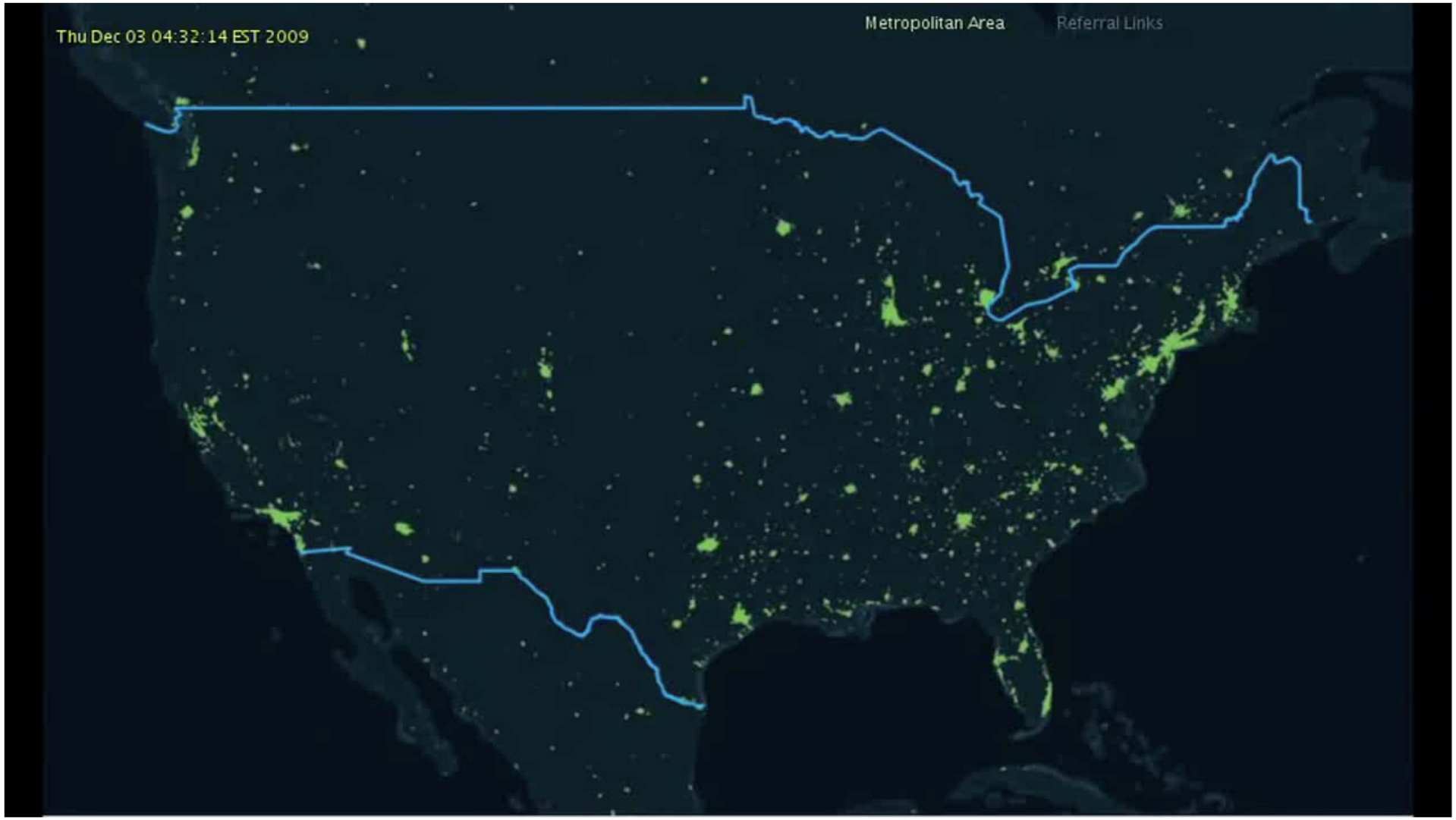


As in many other situations in life, it takes humans to be under pressure to innovate and solve problems otherwise thought unsolvable. And nothing put so much pressure on "social search" scientists than the now-famous DARPA Network Challenge. A $40,000 challenge award would be granted to the first team to submit the locations of 10 moored, eight-foot, red weather balloons at 10 previously undisclosed fixed locations in the continental United States. The balloons were to be placed in readily accessible locations visible from nearby roads. The balloons were deployed at 10:00 AM Eastern Time on December 5, 2009 and scheduled to be taken down at 5:00 PM. DARPA selected the date of the competition to commemorate the 40th anniversary of the Internet. Around 8,000 teams around the world competed in this quest.

Our team at the MIT Media Lab designed a modification of Query Incentive Networks, where the offers were upside down. Instead of the recruiter making an offer to the recruit, it would be the other way around: the recruit would offer a split of the reward back to the recruiter, upon the information being found — *the anti-pyramid scheme*. We called these offers "*split-contracts.*" This modification provided a massive advantage to our team over our competitors, landing us the prize, and capturing worldwide attention, including Stephen Colbert's.



These contracts were quite effective and they were used frequently around the early 2010s. For instance, we used it again to win the Tag Challenge, a transcontinental version of the DARPA Network Challenge, where five simulated jewel thieves where hiding in five different European and North American cities. Participants were able to recruit others in a targeted manner, leading to remarkable convergence towards the target cities. Similarly, they proved to be effective in hybrid versions of social search, such as the Langley Castle Challenge, where some individuals of interest to be found were hiding in the physical world, and others had only a cybernetic presence online. By then, it seemed that social search was pretty much figured out. This collective way of solving problems was even featured on an IBM commercial, as split-


Thu Dec 03 04:32:14 EST 2009
Metropolitan Area
Referral Links

□□□□□□□□□ □ □ □□□□□□□□□□□□□ □□ □□□□□□□□ □□□□□□□□□□t h□□ □t h□□ □□□ □□□ □

Geographic routing in social networks

David Liben-Nowell*†‡§, Jasmine Novak†, Ravi Kumar†¶, Prabhakar Raghavan¶‖, and Andrew Tomkins†¶

PNAS | August 16, 2005 | vol. 102 | no. 33 | 11623–11628

- **We live in a ‘‘small world,’’ where two arbitrary people are likely connected by a short chain of intermediate friends. People can successively forward a message along such a chain.**

- **Existing theoretical models have not been shown to capture behavior in real-world social networks. Here, we introduce a richer model relating geography and social-network friendship, in which the probability of befriending a particular person is inversely proportional to the number of closer people.**

- **In a large social network, we show that one-third of the friendships are independent of geography and the remainder exhibit the proposed relationship. Short chains can be discovered in every network exhibiting the relationship**

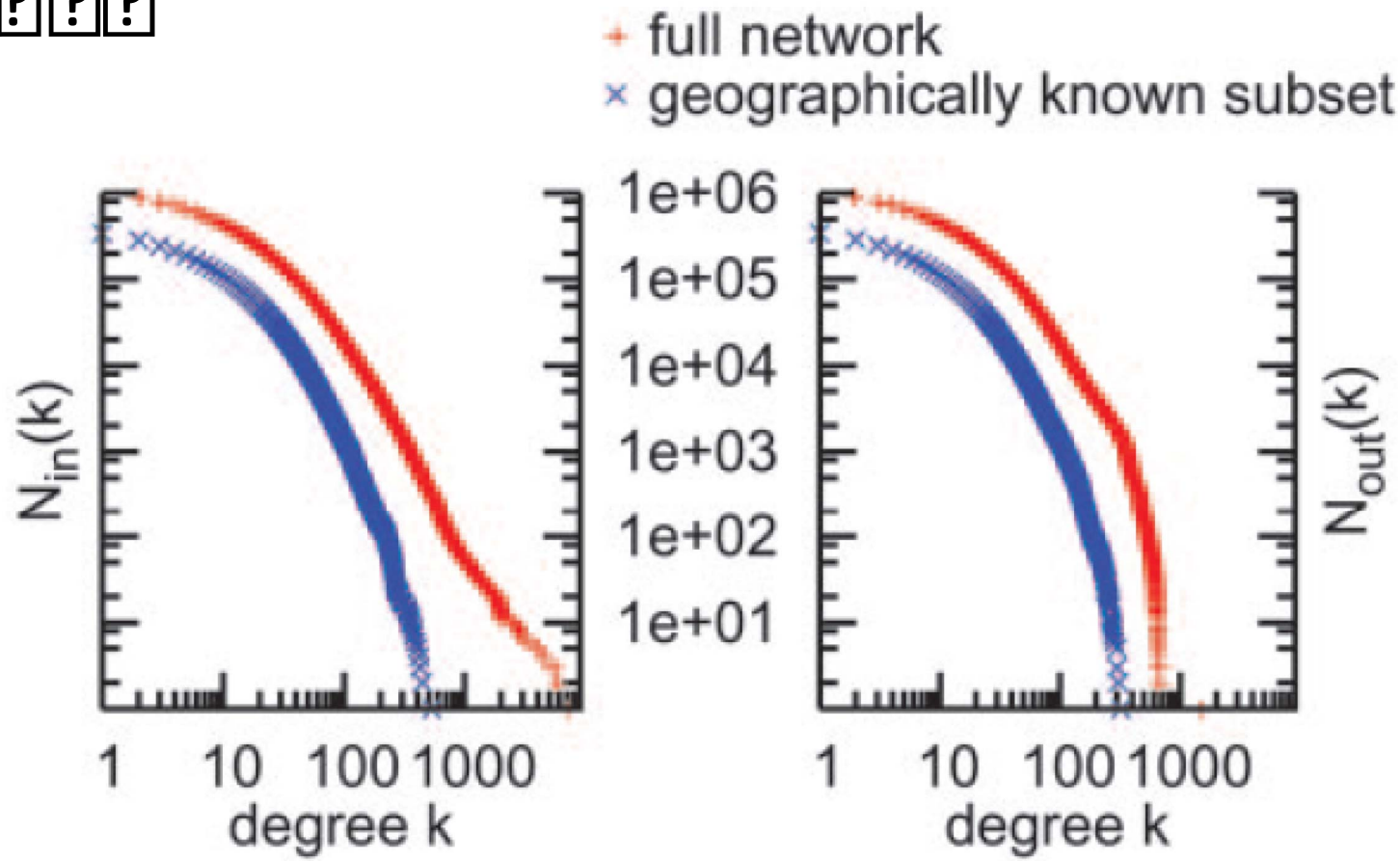


Fig. 1. In-degree (*Left*) and out-degree (*Right*) distributions in LiveJournal. For each k, the number $N_{in}(k)$ of LiveJournal users who are listed as a friend of at least k users and the number $N_{out}(k)$ of people who list at least k friends are shown, both for all 1,300,000 users and the 500,000 users who list locatable hometowns in the United States.

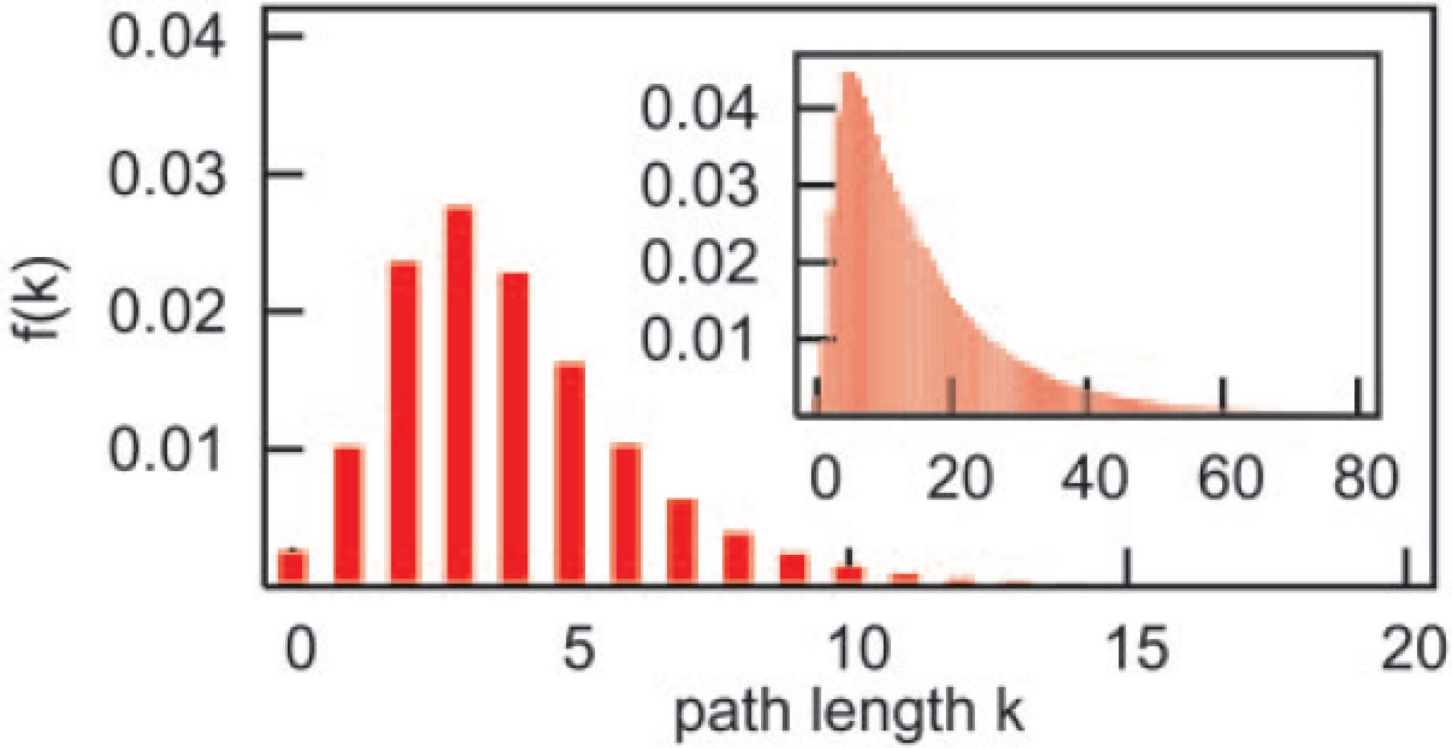
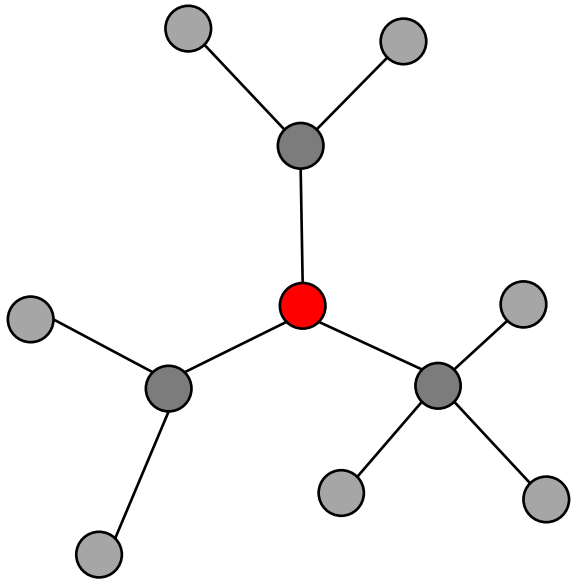


Fig. 2. Results of GEOGREEDY on LiveJournal. In each of 500,000 trials, a source s and target t are chosen randomly; at each step, the message is forwarded from the current message-holder u to the friend v of u geographically closest to t. If $d(v, t) > d(u, t)$, then the chain is considered to have failed. The fraction $f(k)$ of pairs in which the chain reaches t's city in exactly k steps is shown (12.78% chains completed; median 4, $\mu = 4.12$, $\sigma = 2.54$ for completed chains). (*Inset*) For 80.16% completed, median 12, $\mu = 16.74$, $\sigma = 17.84$; if $d(v, t) > d(u, t)$ then u picks a random person in the same city as u to pass the message to, and the chain fails only if there is no such person available.

Random graphs tend to have a tree-like topology with almost constant node degrees.



- nr. of first neighbors: $N_1 \cong \langle k \rangle$
- nr. of second neighbors: $N_2 \cong \langle k \rangle^2$
- nr. of neighbours at distance d: $N_d \cong \langle k \rangle^d$
- estimate maximum distance:

$$N = 1 + \langle k \rangle + \langle k \rangle^2 + \ldots + \langle k \rangle^d = \frac{\langle k \rangle^{d+1} - 1}{\langle k \rangle - 1} \approx \langle k \rangle^d$$

For $r \neq 1$, the sum of the first n terms of a geometric series is

$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = a\,\frac{1 - r^n}{1 - r},$$

$$d = \frac{\log N}{\log \langle k \rangle}$$

Do ClassExercise0322.pdf

Work in group, prepare your questions to the classroom

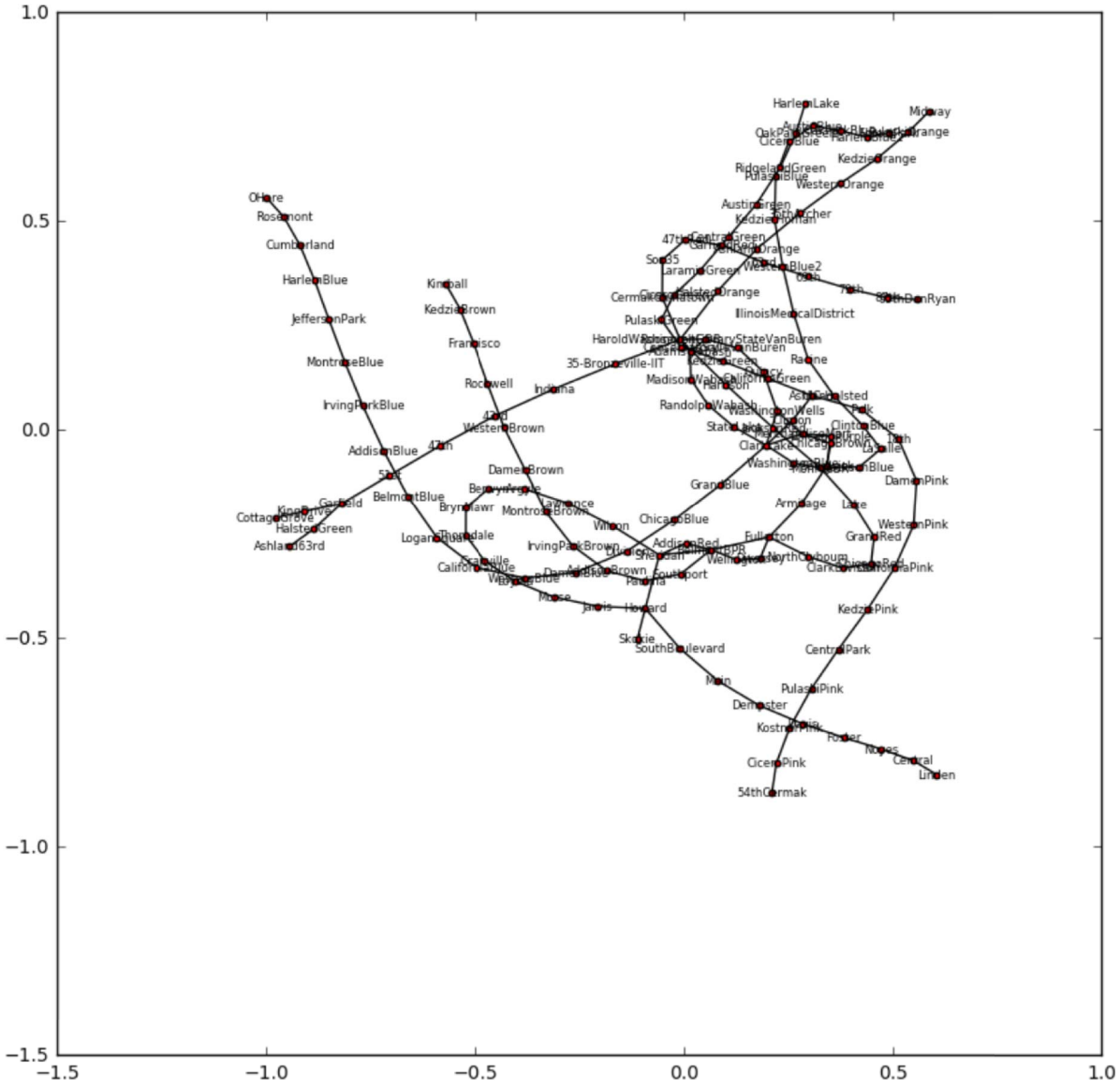
Open Example_SubwayChicago.ipynb

1) Calculate the histogram of the degrees

2) Find the most distant stations (in number of stops)

3) Calculate the histogram of path lengths

□ □□□ □□ □□□□□□□□□□□□ □□□□□




```
g = nx.Graph()
g=nx.read_pajek("Chicago.net")  #read the data
comps = sorted(nx.connected_components(g),key = len, reverse=True)
nodes_gc=comps[0]
```

```
G = nx.subgraph(g,nodes_gc)
```

```
plt.figure(1,figsize=(10,10))
nx.draw_networkx(Gc,node_size=10,font_size=6)
```

```
*Vertices 221
1 "7" 4.050178 124.9847 0.0
2 "14" 4.050178 124.9847 0.0
3 "15" 4.050178 124.9847 0.0
4 "16" 4.050178 124.9847 0.0
5 "20" 4.050178 124.9847 0.0
6 "24" 4.050178 124.9847 0.0
7 "25" 4.050178 124.9847 0.0
8 "26" 4.050178 124.9847 0.0
9 "30" 4.050178 124.9847 0.0
10 "32" 4.050178 124.9847 0.0
11 "40" 4.050178 124.9847 0.0
12 "45" 4.050178 124.9847 0.0
13 "48" 4.050178 124.9847 0.0
14 "54" 4.050178 124.9847 0.0
15 "58" 4.050178 124.9847 0.0
16 "59" 4.050178 124.9847 0.0
17 "60" 4.050178 124.9847 0.0
```

```
214 "OakParkGreen" -2.3481913 -355.26642 0.0
215 "HarlemLake" 1.5083926 -358.0184 0.0
216 "HarlemBlue2" -217.23032 -323.3518 0.0
217 "HarlemBlue" 207.90744 -276.51437 0.0
218 "ForestPark" -218.70464 -327.42108 0.0
219 "Cumberland" 217.45493 -280.2145 0.0
220 "Rosemont" 225.33347 -282.1279 0.0
221 "OHare" 229.88625 -281.31885 0.0
*Edges
137 138 2.0
137 136 2.0
91 110 2.0
91 90 2.0
209 210 2.0
209 206 2.0
176 163 2.0
176 178 2.0
88 89 2.0
```

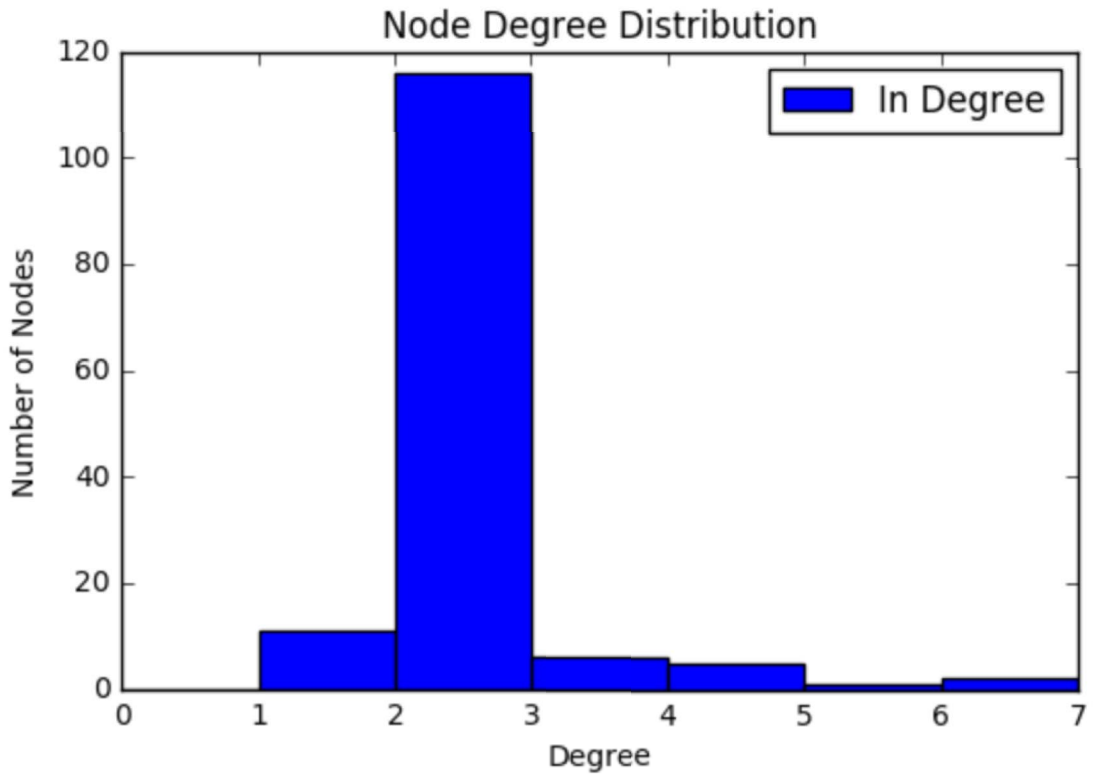
```
:  #array of indegrees, out degrees, and degrees

   deg = dict(G.degree()).values()

   #histogram of each
   hist, bins = np.histogram(deg, bins = range(0,8))
   #hist2, bins = np.histogram(oud, bins = range(0,8))


   #formatting
   width = (bins[1] - bins[0])
   center = (bins[:-1] + bins[1:]) / 2

   #bar graph containing all three
   fig = plt.figure()
   plt.bar(bins[:-1], hist, width = width, color = 'b', label = 'In Degree')
   #plt.bar(bins[:-1] + width, hist2, width = width, color= 'r', label = "Out Degree")
   plt.legend()
   plt.xlabel("Degree")
   plt.ylabel("Number of Nodes")
   plt.title("Node Degree Distribution")
   plt.show()
```



[illegible]

Let's find the maximun of the shortest path lengths

```
lengths = nx.shortest_path_length(G)
st_length = {}  #this will svaes the maximun path per station
ml=0
for key in lengths:   #iterates in all the lengths per station
    ll = key      #only gives an intuitive name
    i = max(ll[1].values())  #finds the maximun value of length
    st_length[ll[0]] = i  #ll[0] has the name of the station and and i the value
    if i > ml:
        ml = i  #this will save the overall maximun
    print "For ", ll[0]," max length is ",i

print "The maximun length in the network is ",ml
```

```
For  Wilson  max length is  24
For  69th  max length is  27
For  PulaskiOrange  max length is  27
For  WesternBrown  max length is  29
```

```
for key in st_length:  #iterates all lengths of each node
    if st_length[key] == ml: #if it equals to the maximun
       print key   #prints the name of the station
       p = nx.shortest_path(g,key)  #calculates all shorthest path from that station
       for k in p:   #iterates all the paths
          if nx.shortest_path_length(g,key,k)==ml: #finds the path that has a length
                                                   #equal to the maximum
             print p[k]   # writes the path
```

```
Kimball
[u'Kimball', u'KedzieBrown', u'Francisco', u'Rockwell', u'WesternBrown', u'DamenBrown', u'Mon
troseBrown', u'IrvingParkBrown', u'AddisonBrown', u'Paulina', u'Southport', u'BelmontBPR', u'
Fullerton', u'Armitage', u'Sedgwick', u'ChicagoPurple', u'MerchandiseMart', u'ClarkLake', u'G
randBlue', u'ChicagoBlue', u'Division', u'DamenBlue', u'WesternBlue', u'CaliforniaBlue', u'Lo
ganSquare', u'BelmontBlue', u'AddisonBlue', u'IrvingParkBlue', u'MontroseBlue', u'JeffersonPa
rk', u'HarlemBlue', u'Cumberland', u'Rosemont', u'OHare']
OHare
[u'OHare', u'Rosemont', u'Cumberland', u'HarlemBlue', u'JeffersonPark', u'MontroseBlue', u'Ir
vingParkBlue', u'AddisonBlue', u'BelmontBlue', u'LoganSquare', u'CaliforniaBlue', u'WesternBl
ue', u'DamenBlue', u'Division', u'ChicagoBlue', u'GrandBlue', u'ClarkLake', u'MerchandiseMart
', u'ChicagoPurple', u'Sedgwick', u'Armitage', u'Fullerton', u'BelmontBPR', u'Southport', u'P
aulina', u'AddisonBrown', u'IrvingParkBrown', u'MontroseBrown', u'DamenBrown', u'WesternBrown
', u'Rockwell', u'Francisco', u'KedzieBrown', u'Kimball']
```

```
lengths = nx.shortest_path_length(G)
plengths=[]
for key in lengths:   #iterates all the kengths
    ll = key
    for i in ll[1].values():    #saves a list with the lengths greater than zero
        if i > 0:plengths.append(i)
```

```
len(plengths)
```

```
19740
```

Possible combinations

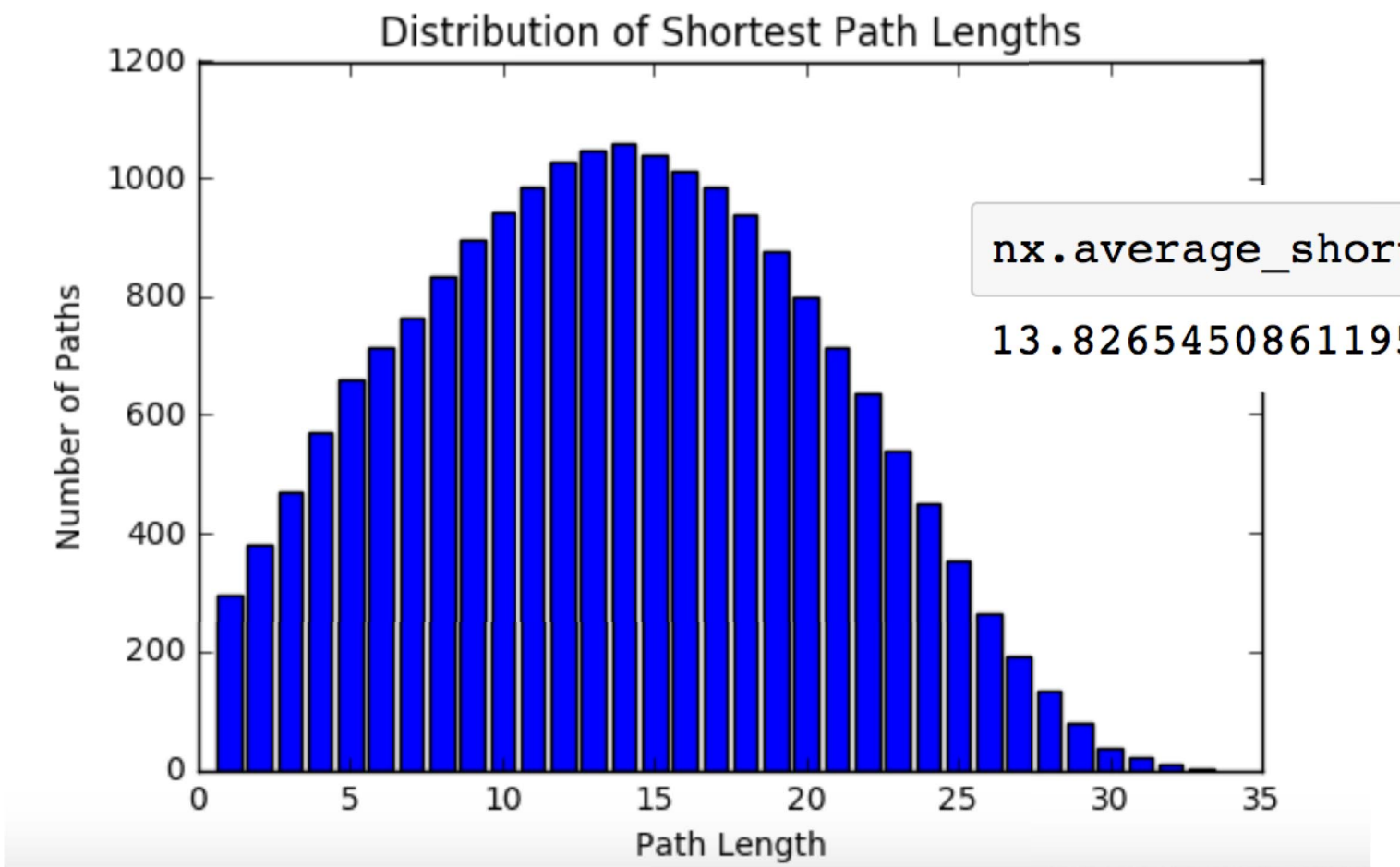
```
G.number_of_nodes()*(G.number_of_nodes()-1)
```

```
19740
```

```
hist4, bins4 = np.histogram(plengths, bins = range(1,35))

plt.bar(bins4[:-1], hist4, align = 'center', color = 'b')

plt.xlabel("Path Length")
plt.ylabel("Number of Paths")
plt.title("Distribution of Shortest Path Lengths")
plt.show()
```




```
nx.average_shortest_path_length(G)
```

```
13.826545086119554
```

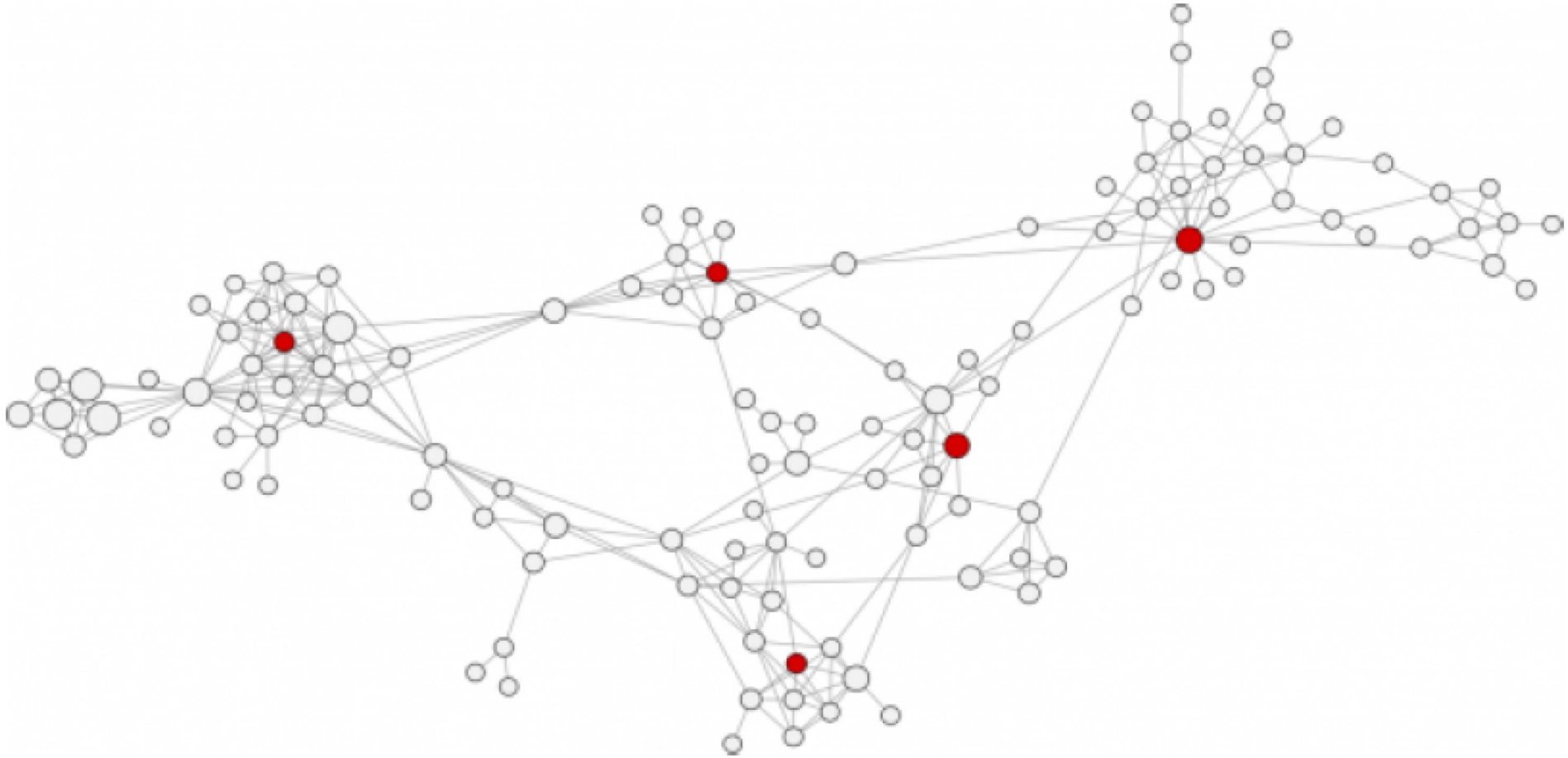
Node Centrality and Link Analysis

Material adapted from lectures of : Leonid E. Zhukov

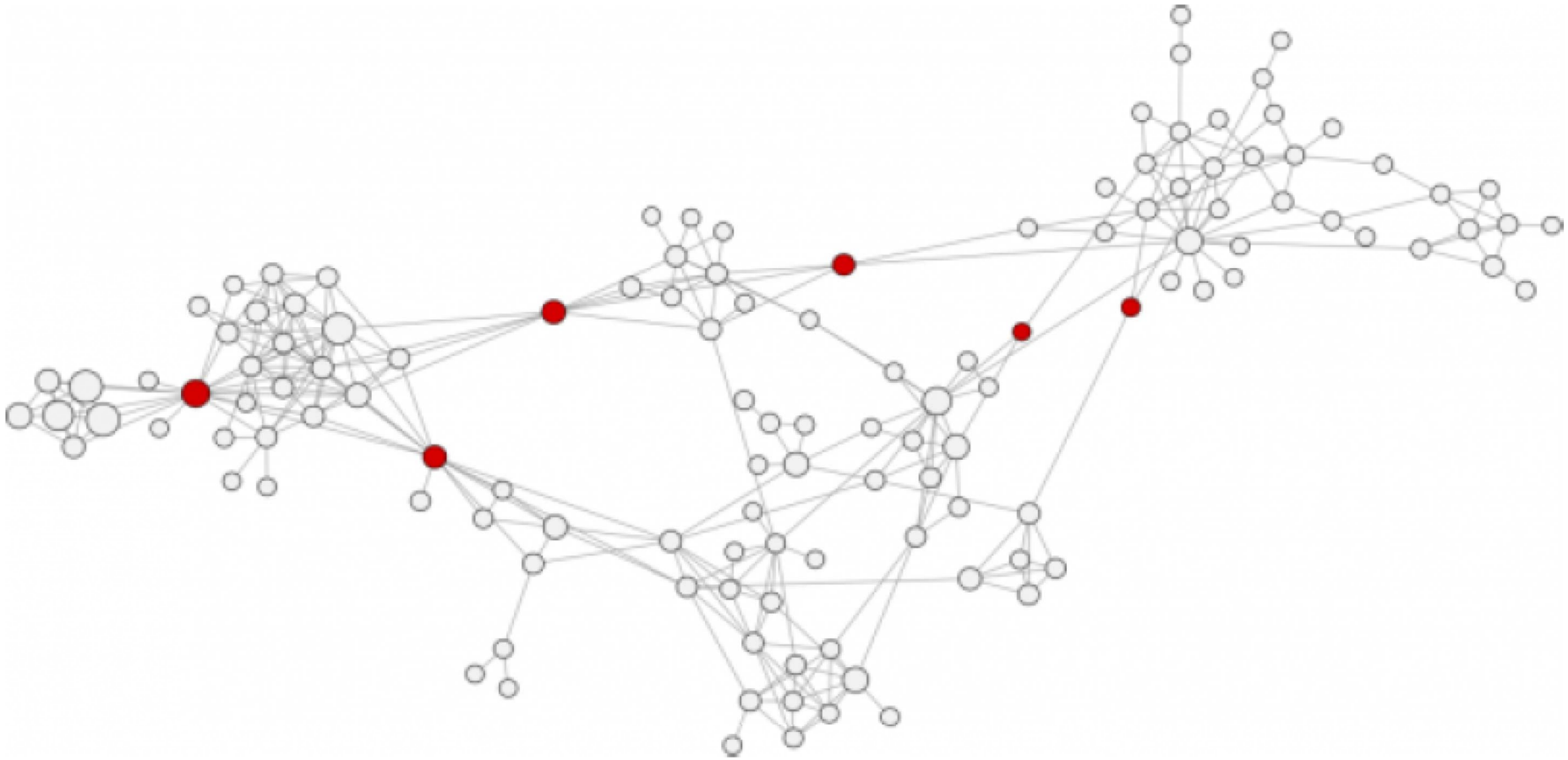
Node Centrality

- Degree Centrality
- Closeness Centrality
- Betewenness Centrality
- Eigenvector Centrality

Closeness centrality



Betweenness centrality



Eigenvector centrality

[illegible]

Degree centrality: number of nearest neighbours

$$C_D(i) = k(i) = \sum_j A_{ij} = \sum_j A_{ji}$$

Normalized degree centrality

$$C_D^*(i) = \frac{1}{n-1} C_D(i)$$

High centrality degree -direct contact with many other actors
Low degree - not active, peripheral actor

[illegible]

Closeness centrality: how close an actor to all the other actors in network

$$C_C(i) = \frac{1}{\sum_j d(i,j)}$$

Normalized closeness centrality

$$C_C^*(i) = (n-1)C_C(i)$$

Actor in the center can quickly interact with all others, short communication path to others, minimal number of steps to reach others

[*** Harmonic centrality $= \sum_j \frac{1}{d(i,j)}$ ***]

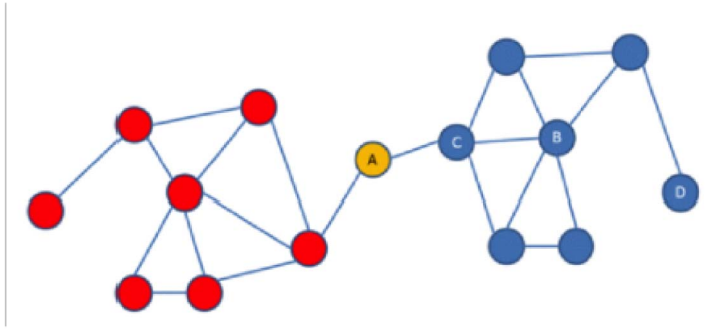
Alex Bavelas, 1948

Betweenness centrality: number of shortest paths going through the actor
$\sigma_{st}(i)$

$$C_B(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

Normalized betweenness centrality

$$C_B^*(i) = \frac{2}{(n-1)(n-2)} C_B(i)$$



Probability that a communication from s to t will go through i

Linton Freeman, 1977

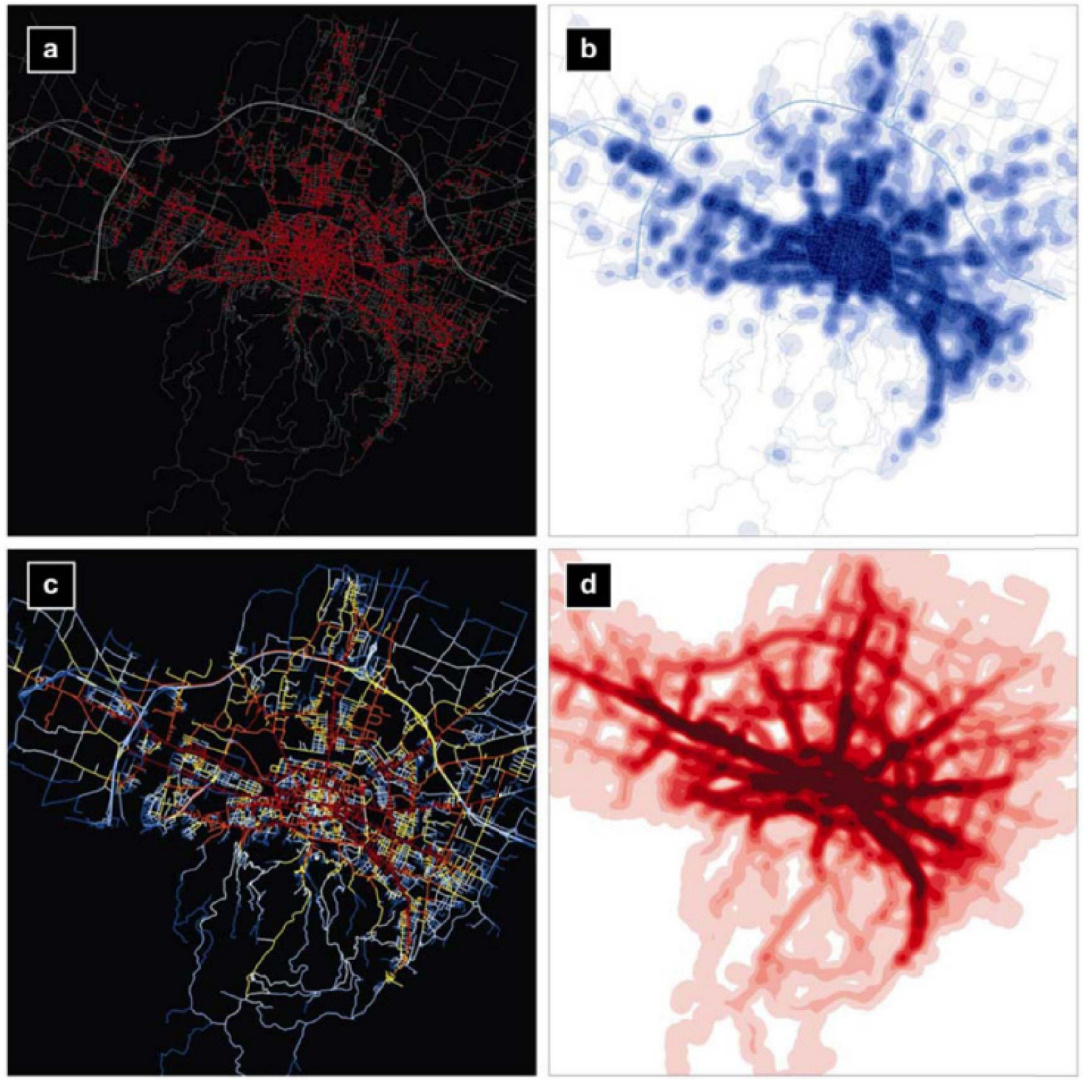


Figure 2. [In colour online, see http://dx.doi.org/10.1068/b34098] Density of activity and street centrality: (a) location of commercial and service activities (red dots); (b) kernel density estimation (KDE) ($h = 300$ m) of commercial and service activities; (c) street global betweenness C^{B}_{glob} (blue for lower values and red for higher); (d) KDE ($h = 300$ m) of C^{B}_{glob}.

Street centrality and densities of retail and services in Bologna, Italy

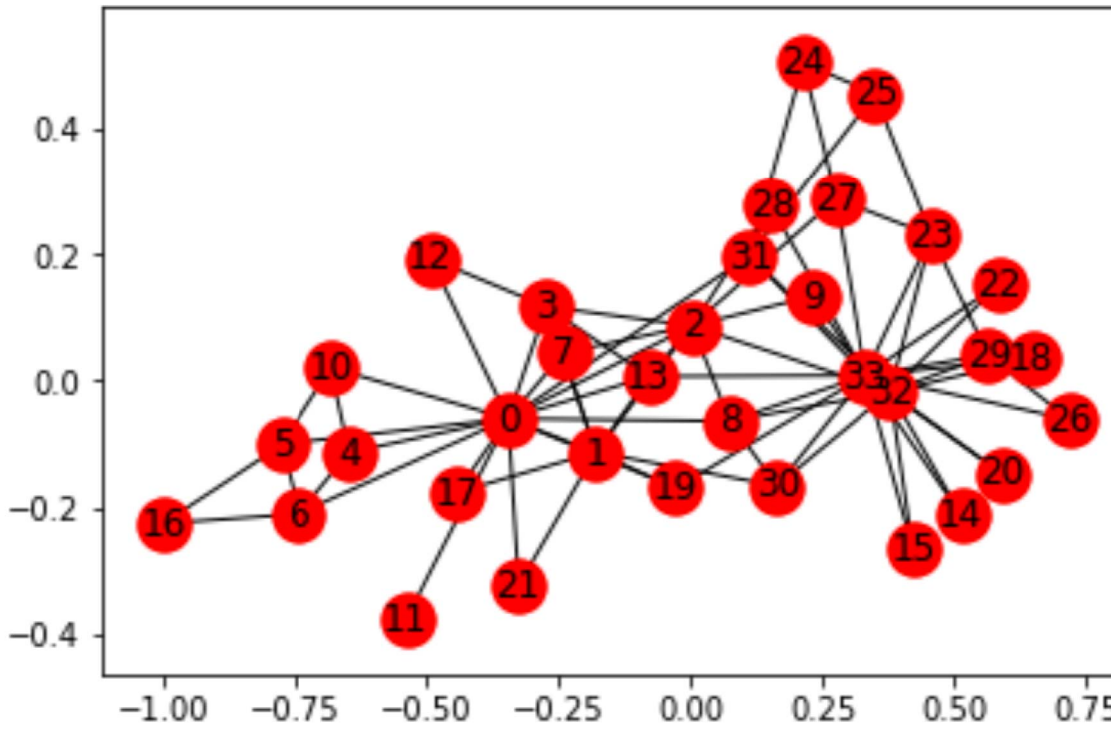
S Porta, E Strano, V Iacoviello, R Messora, V Latora, A Cardillo, F Wang, ...

Environment and Planning B: Planning and design 36 (3), 450-465

???????????

In [2]:
```
G = nx.karate_club_graph()
pos = nx.spring_layout(G) # Fix node positions on all pictures
```

In [3]:
```
# Original network
nx.draw_networkx(G, pos)
```



```python
# Degree centrality
dc = nx.degree_centrality(G)
plt.figure(2, figsize=(7,7))
coord = nx.spring_layout(G)
nx.draw(G,
          pos,
          nodelist=dc.keys(),
          node_size = [d*7000 for d in dc.values()],
          node_color=dc.values(),
          font_size=8,
          cmap=plt.cm.Reds,
          )
```

```python
# Closeness centrality
cl = nx.closeness_centrality(G)
plt.figure(1, figsize=(7,7))
coord = nx.spring_layout(G)
nx.draw(G,
          pos,
          nodelist=cl.keys(),
          node_size = [d*3000 for d in cl.values()],
          node_color=cl.values(),
          font_size=8,
          cmap=plt.cm.Reds,
          )
```