The Importance of Individuals and Groups in Social Networks

By: Saber Gholami
Supervisor: Professor Hovhannes Harutyunyan

Concordia University,
Department of Computer Science and Software Engineering

July 3^{rd}, 2020

Outline

Outline

Introduction

- Social networks are becoming more popular each day!
- Need for studying in a scientific way
- Because of the size:
  - ◇ Solve for specific cases of input,
  - ◇ Working good practically,
  - ◇ Approximation algorithms or,
  - ◇ A mixture.

Outline

Problems in Social Networks

- A Social Network: $G = (V, E)$
- Categories of problems:
  - **Static** vs. Dynamic
  - Content based vs. **Structural**

Table: Instances of Social Networks.

	Weighted	Unweighted
Directed	Email Network	Twitter
Undirected	DBLP	Facebook

Problems in Social Networks - cont.

There are many problems in this area:

- Statistical Analysis:
  - *On a large scale, how does a SN look like?*
  - Extracting statistical features, such as:
    - Degree distribution,
    - Diameter,
    - Clustering behavior,
    - Behavior of connected components,
    - Small-world phenomenon,
    - Power-law degree distribution.

Problems in Social Networks - cont.

- Centrality indices:
  - *How to rank nodes or edges?*
  - Many well-known indices:
    - Degree,
    - Closeness,
    - Betweenness.
  - Due to the size, *estimate* the value.

Problems in Social Networks - cont.

- Centrality indices:
  - *How to rank nodes or edges?*
  - Many well-known indices:
    - Degree,
    - Closeness,
    - Betweenness.
  - Due to the size, *estimate* the value.
- **PageRank and RandomWalks:**
  - Initiate a random surfer in the network,
  - It will end up in most *important* nodes.

Problems in Social Networks - cont.

- **Community Detection:**
  - Members of a community have:
    - Strong connections within the community,
    - Loose connections to those outside.

Problems in Social Networks - cont.

- **Community Detection:**
  - Members of a community have:
    - Strong connections within the community,
    - Loose connections to those outside.
- Influence Maximization:
  - *How does a message spread throughout the network?*
  - 2 important research directions:
    - Model the influence of individuals on each other,
    - Select the best individuals for initiating the spread.

Problems in Social Networks - cont.

- Link Prediction:
  - Which edges are likely to appear in the future?
    - Friends suggestion,
    - Product recommendation,
    - expert hiring.

Problems in Social Networks - cont.

- Link Prediction:
  - Which edges are likely to appear in the future?
    - Friends suggestion,
    - Product recommendation,
    - expert hiring.
- Other problems:
  - Node classification
  - Expert discovery
  - Privacy issues
  - Visualizing
  - Data mining
  - . . .

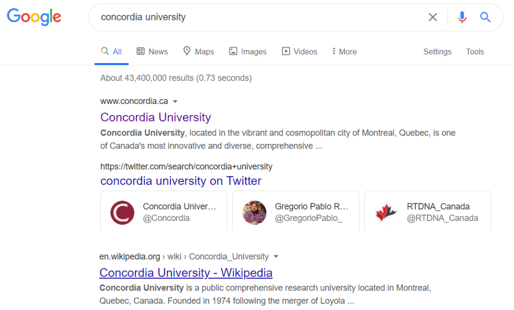
Problems in Social Networks - cont.

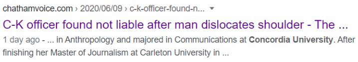
- Link Prediction:
  - Which edges are likely to appear in the future?
    - Friends suggestion,
    - Product recommendation,
    - expert hiring.
- Other problems:
  - Node classification
  - Expert discovery
  - Privacy issues
  - Visualizing
  - Data mining
  - ...
- We choose 2 of them: PageRank and Community Detection.

Outline

Problem Definition

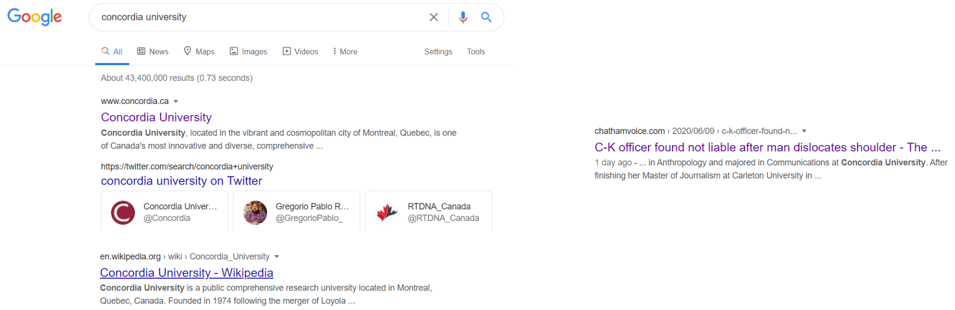
- Motivation: Search Engines





Problem Definition

- Motivation: Search Engines



- IR + PR

Problem Definition - cont.

- Intuition: *The more incoming edges a node has, the more important it is.*
  - ◇ Justification: Good websites do not link many pages, while recieve many links.
- Page P_i with importance r_i has n outgoing edges; each edge gets $\frac{r_i}{n}$.
- Importance of page P_j is the sum of the votes [15]:

$$r_j = \sum_{i \to j} \frac{r_i}{d_i} \tag{1}$$

- Stochastic adjacency matrix M [15]:

$$M_{j,i} = \begin{cases} \frac{1}{d_i} & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

- Assume that $\sum_i r_i = 1$, and $r = [1/N, 1/N, \cdots, 1/N]^T$

Problem Definition - cont.

- Main equation [15]:

$$r = M.r \tag{3}$$

- Power Iteration method [15]:

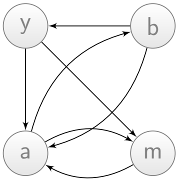
Algorithm 1 Power Iteration

input : Graph G with N nodes, ε
output: PageRank vector r
Initialize $r^{(0)} = [\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}]^T$
while $|r^{(t+1)} - r^{(t)}|_1 < \varepsilon$ **do**
  $r^{(t+1)} = M.r^{(t)}$
end
return r

Problem Definition - cont.

Example:

$$M = \begin{array}{c} \\ a \\ b \\ y \\ m \end{array} \begin{array}{c} \begin{array}{cccc} a & b & y & m \end{array} \\ \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix} \end{array} \quad (4)$$

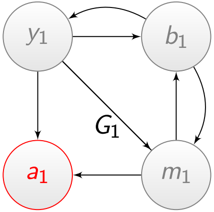


$$r = \begin{array}{c} a \\ b \\ y \\ m \end{array} \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \rightarrow \begin{bmatrix} 1/2 \\ 1/8 \\ 1/8 \\ 1/4 \end{bmatrix} \rightarrow \begin{bmatrix} 6/16 \\ 1/4 \\ 1/16 \\ 5/16 \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} 0.42 \\ 0.21 \\ 0.11 \\ 0.26 \end{bmatrix} \quad (5)$$

Challenges - Dead Ends

- A page with no out going edge is a dead end.
- Surfer gets stuck in this page $\rightarrow$ Page rank will drain out.

$$M = \begin{array}{c} \\ a_1 \\ b_1 \\ y_1 \\ m_1 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & b_1 & y_1 & m_1 \end{array} \\ \begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{3} & 0 \end{bmatrix} \end{array} \qquad (6)$$

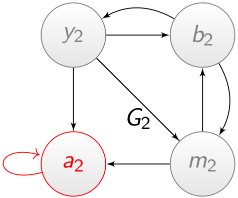


- Solutions:
  - ◇ Removing them,
  - ◇ Teleporting.

Challenges - Spider Traps

- A set of page with no edges outside the set.
- Surfer gets stuck in these pages → They attract all the PageRank.

$$M = \begin{array}{c} \\ a_2 \\ b_2 \\ y_2 \\ m_2 \end{array} \begin{array}{c} \begin{matrix} a_2 & b_2 & y_2 & m_2 \end{matrix} \\ \begin{bmatrix} 1 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{3} & 0 \end{bmatrix} \end{array} \quad (7)$$



- Solutions:
  - ◇ Removing them,
  - ◇ Teleporting.

Dealing with challenges: Teleporting

The random surfer has 2 options:

- With probability β follow an edge,
- With probability $1-\beta$ jump to a random page.

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N} \quad (8)$$

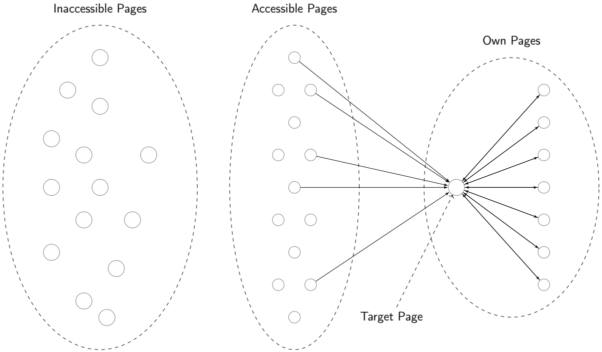
- Matrix M is modified:

$$A = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N} \quad (9)$$

- β is normally between 0.8 and 0.9.
- + Always teleport from a dead end!

Link Spam

- Not all individuals behave well in the WWW network.
- Designing a structure to increase the PR of a page, artificially.
- Architecture of a Spam Farm:



Link Spam - cont.

- How to deal with Spam Farm?
  - Detect these structures,
  - Modify Page Rank:
    - Gyongyi et al. [10], Trust Rank: Teleport to trustworthy pages,
    - Gyongyi et al. [9], Spam Mass: $\frac{r-t}{r} \approx 1 \rightarrow$ SPAM!

Literature Review - Spam Farm

- Wu et al. [22]: Pages in a Spam Farm are densely connected;
  - Seed set of bad pages: more than T_{IO} common in-out links,
  - Pages that link to more than T_{PP} bad pages: potential spammers,
  - Give 0 weight to links between bad pages,
  - Recalculate PR.

Literature Review - Spam Farm - cont.

- Ghosh et al. [7], Collusionrank: Penalize the ones who are following *bad* users:

Algorithm 2 Collusionrank

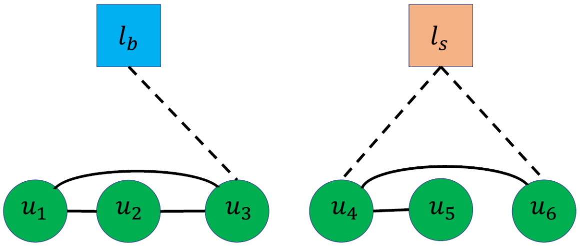
input : Graph G, Set of known spammers S, β
output: Collusionrank vector c

$$d(n) = \begin{cases} \frac{-1}{|S|} & \text{if } n \in S, \\ 0 & \text{otherwise} \end{cases}$$

c $\leftarrow d$
while *c not converged* **do**
  foreach *node v in G* **do**
    $tmp \leftarrow \sum_{n \in following(v)} \frac{c(n)}{|followers(n)|}$
    $c(n) = \beta \times tmp + (1 - \beta) \times d(n)$
  end
end
return c

Literature Review - Spam Farm - cont.

- Jia et al. [13]:
  - Two artificial nodes: real l_b, fake l_s,
  - Add edge from every fake known node to l_s,
  - Add edge from every real known node to l_b,
  - For each unknown node, initiate a RandomWalk,
  - Badness score = the probability of reaching l_s sooner than l_b.



Literature Review - Applications of PageRank

- In link prediction problem:
  - Tong et al. [21]:
    - Initiate a random walk with restart from v,
    - Nodes with highest PR score will form an edge with v.
  - Backstrom and Leskovec [2]:
    - Initiate a random walk from v,
    - Learn to visit the nodes that will have a potential edge.

Literature Review - Applications of PageRank - cont.

- In Influence Maximization problem:
  - Java et al. [11]:
    - Nodes with high PR are good seed sets
  - Bar-Yossef et al. [3], Reverse PageRank:
    - Change the direction of edges
    - Run PageRank
    - Nodes with high PR are good seed sets

Outline

Problem Definition

- In graph $G = (V, E)$, divide V into c subsets, $C_1, C_2, \cdots, C_c$ in a way that:
  - Members of each community have dense connections inside,
  - And loose connections to those outside
- In 2 phases:
  - *Detecting* communities with an algorithm,
  - *Evaluating* the appropriateness of communities.

Taxonomy of Methods

- 4 Categories of methods:
  - Disjoint or non-overlapping communities,
  - Overlapping communities,
  - Hierarchical communities and,
  - Local communities.
- Evaluation Metrics

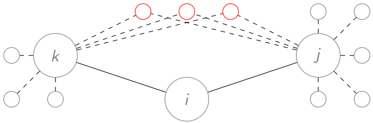
- Girvan and Newman [8] ($O(VE^2)$ or $O(V^3)$):
  - Calculate edge betweenness for all edges,
  - Remove the edge with highest betweenness,
  - Recalculate the edge betweenness for all edges,
  - Repeat step 2 and 3 until no edge remains.

Taxonomy of Methods - Overlapping Communities

- Ahn et al. [1], Link algorithm:
  - Similarity of two edges:

$$S(e_{ij}, e_{ik}) = \frac{|n_+(j) \cap n_+(k)|}{|n_+(j) \cup n_+(k)|} \tag{10}$$

  - Merge the edges with highest similarity into 1 community,
  - For e_{ij} and e_{ik}, if k and j belong to different communities, i is an overlapping node.



Taxonomy of Methods - Hierarchical Communities

- Mann et al. [18]:
  - Using the idea of sparsest cut:

$$(S, T) - \text{cut density} = \frac{|(S, T)|}{(|S| \cdot |T|)} \tag{11}$$

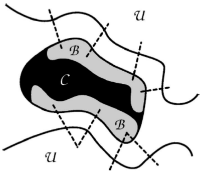
  - The cut with minimum density is suitable for partitioning the graph,
  - Find minimum density cut and repeat it for bigger sub-graph.
  - Finding sparsest cut is NP-Hard:
    - Closely related to maximum concurrent flow,
    - Could be solved efficiently with linear programming.

Taxonomy of Methods - Local Communities

- Clauset [5]:
  - Divide V into three sets: $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{U}$.
  - Local modularity:

$$R = \frac{I}{T} \tag{12}$$

    - I: the number of those edges with neither end point in $\mathcal{U}$
    - T: the number of edges with one or more end points in $\mathcal{B}$
  - Start with $C = v_0$ and discover k vertices that are in the same community as v_0,
  - In each step, add the one with highest difference in terms of R.



Taxonomy of Methods - Evaluation Metrics

- Luo et al. [16]:

$$M = \frac{E_{in}}{E_{out}} \tag{13}$$

  - E_{in}: Number of edges within the community,
  - E_{out}: Number of crossing edges.

- Chen et al. [4]:

$$L = \frac{L_{in}}{L_{out}} \tag{14}$$

  - $L_{in} = \frac{E_{in}}{|\mathcal{C}|}$
  - $L_{out} = \frac{E_{out}}{|\mathcal{B}|}$

Outline

Conclusion and Future Work

- 2 important problems in social networks were considered in this study:
  - PageRank and RandomWalks
  - Community Detection
- We're interested in the following directions for future work:
  - Connection between PageRank and Broadcasting
  - Local community detection
    - With PageRank
    - Evaluation Metric

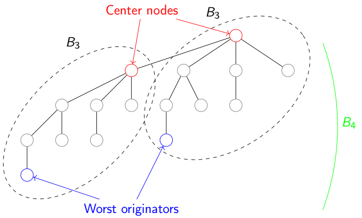
Conclusion and Future Work - cont.

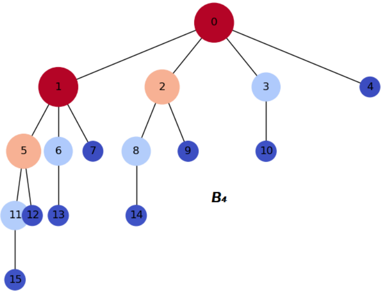
1. Connection between PageRank and Broadcasting:
   - Broadcasting:
     - ◇ A message is transmitted throughout the network,
     - ◇ Starting from a single *originator*,
     - ◇ All informed vertices may initiate a call in each time step,
     - ◇ $b(G) = \max_{v \in V}\{b(v, G)\}$
     - ◇ $\lceil \log_2 n \rceil \leq b(G) \leq n - 1$
     - ◇ Research directions:
       - · *mbg*s
       - · Find broadcast time of any graph: NP-Complete
       - ✓ Find Center nodes
       - ✓ Find Worst originators
   - Experiment: Run Power Iteration method with $\beta = 0.85$ and $\varepsilon = 0.00005$

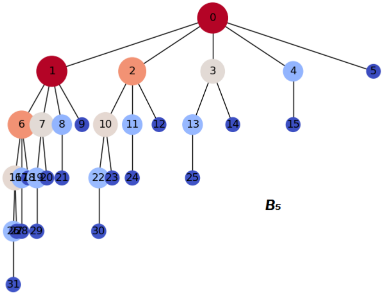
Conclusion and Future Work - cont.

1. Connection between PageRank and Broadcasting:

- Binomial Tree:

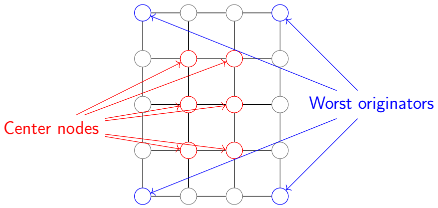


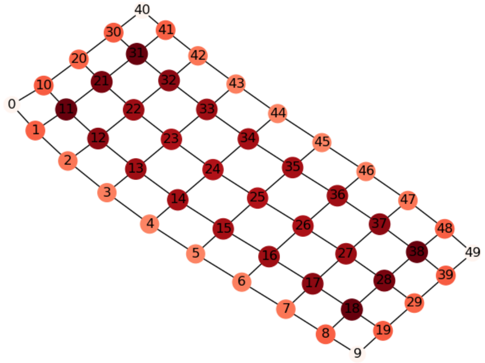




Conclusion and Future Work - cont.

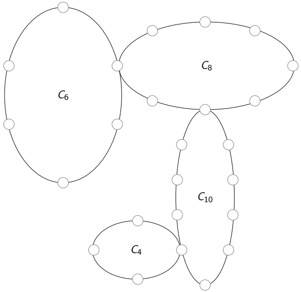
1. Connection between PageRank and Broadcasting:
   - Grids:

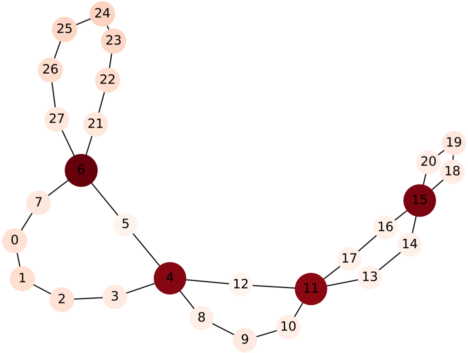




Conclusion and Future Work - cont.

1. Connection between PageRank and Broadcasting:
   - Necklace graph:





Conclusion and Future Work - cont.

2.1. Local Community Detection - Method:

- Source vertex v_0,
- Discover k vertices that are in the same local community as v_0
- Possible algorithm:
  - Start a random walk from v_0 (with TrustRank modification)
  - $C = v_0$, Trusted$= v_0$
  - Until $|C| = k$:
    - Add v_i with the highest page rank to the community: $C = C \cup v_i$
    - Trusted $=$ Trusted $\cup v_i$
- Why it works?
  - The vertices with highest PageRank have good similarity with v_0
  - In $1 - \beta$ fraction of times, teleport to the discovered community

Conclusion and Future Work - cont.

2.2. Local Community Detection - Evaluation Metric:

- Idea of using Geodesic Distance (GD),
- Length of the shortest path between two nodes,
- Possible metric:
  - Sum of GD for all vertices within a local community,
  - The smaller the sum, the better the community.
- Why it works?
  - More edges in a community $\rightarrow$ length of shortest path will decrease,
  - More edges in a community $\rightarrow$ the community is more dense.
- Also, design a heuristic algorithm that uses this metric:
  - Add the vertices with highest difference in terms of the metric to local community.

Important References I

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann.
Link communities reveal multiscale complexity in networks.
nature, 466(7307):761–764, 2010.

[2] Lars Backstrom and Jure Leskovec.
Supervised random walks: predicting and recommending links in social networks.
In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644, 2011.

[3] Ziv Bar-Yossef and Li-Tal Mashiach.
Local approximation of pagerank and reverse pagerank.
In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 279–288, 2008.

[4] Jiyang Chen, Osmar Zaïane, and Randy Goebel.
Local community identification in social networks.
In *2009 International Conference on Advances in Social Network Analysis and Mining*, pages 237–242. IEEE, 2009.

Important References II

[5] Aaron Clauset.
Finding local community structure in networks.
Physical review E, 72(2):026132, 2005.

[6] Imre Derényi, Gergely Palla, and Tamás Vicsek.
Clique percolation in random networks.
Physical review letters, 94(16):160202, 2005.

[7] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi.
Understanding and combating link farming in the twitter social network.
In *Proceedings of the 21st international conference on World Wide Web*, pages 61–70, 2012.

[8] Michelle Girvan and Mark EJ Newman.
Community structure in social and biological networks.
Proceedings of the national academy of sciences, 99(12):7821–7826, 2002.

Important References III

[9] Zoltan Gyongyi, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen.
Link spam detection based on mass estimation.
In *Proceedings of the 32nd international conference on Very large data bases*, pages 439–450. VLDB Endowment, 2006.

[10] Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen.
Combating web spam with trustrank.
In *Proceedings of the 30th international conference on very large data bases (VLDB)*, 2004.

[11] Akshay Java, Pranam Kolari, Tim Finin, and Tim Oates.
Modeling the spread of influence on the blogosphere.
UMBC TR-CS-06-03, 2006.

[12] Glen Jeh and Jennifer Widom.
Simrank: a measure of structural-context similarity.
In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002.

Important References IV

[13] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong.
Random walk based fake account detection in online social networks.
In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 273–284. IEEE, 2017.

[14] Amy N Langville and Carl D Meyer.
Deeper inside pagerank.
Internet Mathematics, 1(3):335–380, 2004.

[15] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman.
Mining of massive data sets.
Cambridge university press, 2020.

[16] Feng Luo, James Z Wang, and Eric Promislow.
Exploring local community structures in large networks.
In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 233–239. IEEE, 2006.

Important References V

[17] Wenjian Luo, Daofu Zhang, Li Ni, and Nannan Lu.
Multiscale local community detection in social networks.
IEEE Transactions on Knowledge and Data Engineering, 2019.

[18] Charles F Mann, David W Matula, and Eli V Olinick.
The use of sparsest cuts to reveal the hierarchical community structure of social networks.
Social Networks, 30(3):223–234, 2008.

[19] Mark EJ Newman.
Fast algorithm for detecting community structure in networks.
Physical review E, 69(6):066133, 2004.

[20] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara.
Near linear time algorithm to detect community structures in large-scale networks.
Physical review E, 76(3):036106, 2007.

Important References VI

[21] Hanghang Tong, Christos Faloutsos, and Yehuda Koren.
Fast direction-aware proximity for graph mining.
In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 747–756, 2007.

[22] Baoning Wu and Brian D Davison.
Identifying link farm spam pages.
In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 820–829, 2005.

[23] Ye Zhen-Qing, Zhang Ke, Hu Song-Nian, and Yu Jun.
A new definition of modularity for community detection in complex networks.
Chinese Physics Letters, 29(9):098901, 2012.

Thanks a bunch!

Appendix-PageRank works

Why power iteration method works?

- Recall that when $Ax = \lambda x$, x is the eigenvector and λ is the eigenvalue.
- In equation $r = M.r$: r is the principal eigenvector of M with eigenvalue of 1 (largest eigenvalue of M)
    - Because M is column stochastic.
- Eventually, we want to find the dominant eigenvector of M.
- $r^{(1)} = M.r^{(0)}$
- $r^{(2)} = M.r^{(1)} = M(M.r^{(0)}) = M^2.r^{(0)}$
- $\cdots$
- $r^{(k)} = M^k.r^{(0)}$

Claim

The sequence of $M.r^{(0)}, M^2.r^{(0)}, \cdots, M^k.r^{(0)}$ approaches the dominant eigenvector of M.

Appendix-PageRank works

Claim

The sequence of $M.r^{(0)}, M^2.r^{(0)}, \cdots, M^k.r^{(0)}$ approaches the dominant eigenvector of M.

Proof.

Suppose M has n eigenvectors $x_1, x_2, \cdots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$ in a way that: $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ [15].

We can write $r^{(0)} = c_1x_1 + c_2x_2 + \cdots + c_nx_n$, now:

$$M.r^{(0)} = M(c_1x_1 + c_2x_2 + \cdots + c_nx_n) \rightarrow$$

$$M.r^{(0)} = c_1(Mx_1) + c_2(Mx_2) + \cdots + c_n(Mx_n) \xrightarrow{M.x_i=\lambda_i.x_i}$$

$$M.r^{(0)} = c_1(\lambda_1x_1) + c_2(\lambda_2x_2) + \cdots + c_n(\lambda_nx_n) \xrightarrow{\text{repeat multiplication on both sides}}$$

$$M^k.r^{(0)} = c_1(\lambda_1^kx_1) + c_2(\lambda_2^kx_2) + \cdots + c_n(\lambda_n^kx_n) \rightarrow$$

$$M^k.r^{(0)} = \lambda_1^k(c_1x_1 + c_2(\frac{\lambda_2}{\lambda_1})^kx_2 + \cdots + c_n(\frac{\lambda_n}{\lambda_1})^kx_n) \xrightarrow[\lim_{k\to\infty}(\frac{\lambda_i}{\lambda_1})^k=0]{\lambda_1>\lambda_2>\cdots>\lambda_n}$$

$$M^k.r^{(0)} = c_1(\lambda_1^kx_1)$$

Appendix-PageRank works

A note on β:

- We want to simulate the user's behavior. With $\beta = 0.85$, we are giving a chance of entering a new URL in $\frac{1}{6}$ of times.
- As β increases, the PageRank becomes more and more sensitive to small changes in M matrix.
- The smaller the β, the faster the convergence, but the structure of the graph is not used so much!
- A trade off!
- Langville [14] showed that a rough estimate of the number of iterations needed to converge to a tolerance level ε is $\frac{\log_{10}\varepsilon}{\log_{10}\beta}$, So:
  - ◇ for $\beta = 0.85$ and $\varepsilon = 10^{-6}$ it takes roughly $\frac{-6}{\log_{10}0.85} \approx 85$ (A very common situation),
  - ◇ Or for $\beta = 0.85$ and $\varepsilon = 10^{-8}$ it takes almost 114 iterations,
  - ◇ While for $\beta = 0.99$ and $\varepsilon = 10^{-8}$, it takes 1833 iterations!

Appendix-PageRank works

Tong et al. [21]:

- Proposing node-to-node proximity measure Prox based on RandomWalks,
- escape probability $ep_{i,j}$: the probability that the random walk which starts from node i will visit node j before it returns to node i,
- generalized voltage $v_k(i,j)$: the probability that a random walk that starts from node k will visit node j before node i,
- $p_{i,k}$: probability of a direct transition from node i to node j,
- Prox measure: $\text{Prox}(i,j) \triangleq ep_{i,j} = \sum_{k=1}^{n} p_{i,k}.v_k(i,j)$
- **Predict a link** between i and j iff $\text{Prox}(i,j) + \text{Prox}(j,i) > th$,
    - ◇ th is a given threshold.

Appendix-PageRank works

Backstrom and Lescovec [2]:

- Combining two general approaches for link prediction:
  - ◇ Using graph structural information (with RandomWalk),
  - ◇ Using node and edge attributes (with ML).
- Assign each edge a RandomWalk transition probability (learn strength function for each edge),
- Initiate a RandomWalk with restart from source node s,
- Nodes with highest PageRank are the ones that s will form an edge with.
- Excellent results on co-authorship network $\rightarrow$ suggest who to write a paper with!

Appendix-PageRank works

Jeh et al [12]:

- Intuition: two objects are similar if they are related to similar objects,
- SimRank: if $a = b$ then $s(a, b) = 1$. Otherwise:

$$s(a,b) = \frac{C}{|\text{InDegree}(a)|.|\text{InDegree}(b)|} \sum_{v \in \text{InNeighbor}(a)} \sum_{u \in \text{InNeighbor}(b)} s(u,v) \tag{15}$$

- They show that SimRank score $s(a, b)$ measures how soon two random surfers are expected to meet at the same node if they started at nodes a and b and randomly walked the graph backwards.

$$m(a,b) = \sum_{t:(a,b) \rightsquigarrow (x,x)} P[t] l(t) \tag{16}$$

  - ◇ $t =< w_1, \cdots, w_k >$ is a tour on G^2 graph with V^2 as the nodes and $< (a, b), (c, d) > \in E(G^2)$ means $(a, c) \in E(G)$ and $(b, d) \in E(G)$
  - ◇ $P[t]$ is the probability of traveling $P[t] = \prod_{i=1}^{k-1} \frac{1}{|\text{OutDegree}(w_i)|}$
  - ◇ $l(t)$ is the path length and is $k - 1$.

Appendix-PageRank works

Bar-Yossef et al [3]:

- They show that local PageRank approximation is not efficient in graphs with high in-degree nodes (such as SNs).
- However, ReversePageRank can be approximated locally in the graph obtained by reversing the direction of all edges.
- They also argue that ReversePageRank is useful for selecting influential nodes in IM problem, and many other applications (such as crawler's seed set selection).

Appendix-Community Detection works

Fast Newman [19]:

- $O(V(V+E))$ or $O(V^2)$:
  - ◇ Modularity $Q = \sum_i (e_{ii} - a_i^2)$:
    - e_{ii} fraction of edges within the group i
    - e_{ij} one-half of the fraction of edges that connect a vertex from group i to j
    - $a_i = \sum_j e_{ij}$ the fraction of all ends of edges that are attached to vertices in the group i
    - a_i^2 the value that it would take if edges were placed at random.
  - ◇ Two nodes are joined with biggest difference in Q.

Appendix-Community Detection works

Fast Newman [19]:

- Modularity $Q = \sum_i (e_{ii} - a_i^2)$
- Why not optimize Q over all possible divisions to find the best one?
- It is very costly:
  - ◇ Number of ways to divide n objects into g non-empty groups is the Stirling number of the second kind $S_n^{(g)}$,
  - ◇ The sum is not known in closed form,
  - ◇ But we know that $S_n^{(1)} + S_n^{(2)} = 2^{n-1}$; thus, it is at least exponentially!
- Instead, use a greedy approximation optimization:
  - ◇ Each vertex is the sole member of a community,
  - ◇ Repeatedly join communities together,
  - ◇ Choosing the join that results in the greatest increase (or smallest decrease) in Q.
  - ◇ Generates a dendrogram!

Appendix-Community Detection works

- Derènyi et al. [6], Clique Percolation Method (CPM):
  - Find all *k-cliques*,
  - Build hyper graph (two k-cliques are connected if they share k-1 vertices),
  - Connected parts are communities.

Appendix-Community Detection works

- Raghavan et al. [20], Label Propagation Algorithm (LPA):
    - The community of x is the same as majority of its neighbors,
    - Initiate labels,
    - Propagate until convergence.
    - BUT it is likely to find many communities for the same graph.

Appendix-Community Detection works

Ahn et al. [1]:

- Where to cut the generated dendrogram?
- They proposed partition density D as follows:
  - ◇ For a graph with M edges and N nodes, $P = \{P_1, \cdots, P_C\}$ is a partition of the links into C subsets.
  - ◇ The number of links in subset P_c is m_c,
  - ◇ The number of adjacent nodes in subset P_c is $n_c = |\cup_{e_{i,j} \in P_c} \{i, j\}|$
  - ◇ Density of community c is:

$$D_c = \frac{m_c - (n_c - 1)}{\frac{n_c(n_c - 1)}{2} - (n_c - 1)} \tag{17}$$

  - ◇ The partition density D is the average of D_c weighted by the fraction of present links:

$$D = \frac{2}{M} \sum_c m_c \frac{m_c - (n_c - 1)}{(n_c - 2)(n_c - 1)} \tag{18}$$

- Cut dendrogram where maximum D happens.

Appendix-Community Detection works

Mann et al. [18]:

- Using the idea of the sparsest cut (highly related to MCFP).
- Ford-Fulkerson method:
  - While there is an augmenting path in the graph;
  - Augment the value of flow along the path,
  - Reduce the capacities along the path,
- If the augmenting path is found via BFS, the algorithm is called Edmonds-Karp.
- A good heuristic for finding the sparsest cut.

Appendix-Community Detection works

- Lue et. al [17]:

$$LQ = \frac{e_c}{S} - (\frac{d_c}{2S})^2 \tag{19}$$

  - e_c: number of edges within the detected local community,
  - d_c: summation of degrees of all nodes belonging to that local community,
  - S: the number of edges with one or two endpoints in the local community.
- Zhen-Qing et al. [23]:

$$Q^d = \sum_{r=1}^{s} (\frac{L_r}{D_r} - \frac{\tilde{L}_r}{\tilde{D}_r}) \tag{20}$$

  - L_r: Number of edges inside the community
  - D_r: Average minimal path for all pairs of nodes within a given community,
  - $\tilde{L}_r$ and $\tilde{D}_r$: Expected values for the graph that is generated randomly.

Appendix-Community Detection works

Zhen-Qing et al. [23]:

- We know what are L_r and D_r in this equation $Q^d = \sum_{r=1}^{s}(\frac{L_r}{D_r} - \frac{\tilde{L_r}}{\tilde{D_r}})$
- But how to calculate $\tilde{L_r}$ and $\tilde{D_r}$?
  - $\tilde{L_r} = d_r^2/4L$ where:
    - d_r is the sum degree of nodes in community r,
    - L is the total number of edges for the underlying network.

Appendix-Community Detection works

◇ $\tilde{D}_r$:

- $K = (k_1, k_2, \cdots, k_n)$ is the degree distribution of the original graph,
- L_{ij} The path length of certain vertices (i, j) can be approximately calculated based on

$$L_{ij}(k_i, k_j) = \frac{-\ln k_i k_j + \ln(< k^2 > - < k >) + \ln N - \gamma}{\ln(< k^2 > / < k > -1)} + \frac{1}{2} \quad (21)$$

- $< . >$ indicates the average operation over the entire degree sequence,
- N is the number of vertices,
- γ is a constant value of 0.5772

$$\tilde{D}_r = \frac{2}{n_r(n_r - 1)} \sum_{i,j \in r, i<j} L_{ij}(k_i, k_j) \quad (22)$$

- n_r is the number of nodes in community r.