Distance Degree Sequences for Network Analysis

Christian Pich

Universität Konstanz
Computer & Information Science
Algorithmics Group

15 Mar 2005

based on

Palmer, Gibbons, and Faloutsos: ANF – A Fast and Scalable Tool for Data Mining in Massive Graphs, SIGKDD 02.

Intro
- Motivation
- Foundations
- Applications

ANF Algorithm
- Exact Algorithm
- Approximation Algorithm
- Benefits

Applications
- Web Mining
- Graph Similarity
- Internet Router Data

Summary

Graphs

Problems modeled as graphs appear in various ares, including:

- social networks
- streets
- academic citations
- biology and chemistry
- the Internet
- ...

Questions

Some related questions in network analysis:

- How robust is a network to failures?
- Are two given networks similar?
- Given two actors in a network, which one is more influential?

Typical networks to be analyzed are LARGE

Key issue: Extract a small set of features that describe much of the character of particular actors or the overall network

Definitions I – Basics

- Graph $G = (V, E)$, $E \subseteq V \times V$ (or $\binom{V}{2}$ if G is undirected)
- $n = |V|, m = |E|$
- v, w adjacent $\Leftrightarrow (v, w) \in E$ (or $\{v, w\} \in E$)
- Neighborhood $\text{Neigh}(v) = \{w \in V : (v, w) \in E\}$
- Degree $\deg(v) = |\text{Neigh}(v)|$
- Distance $d(v, w) =$ length of shortest path from v to w
- Diameter $\text{diam}(G) =$ longest distance in a graph (over all $v, w \in E$)

Definitions II – Neighborhoods

- h-neighborhood $\text{Neigh}_h(v) = \{w \in V : d(v,w) \leq h\}$
- $\text{Neigh}_0(v) = \{v\}, \text{Neigh}_1(v) = \text{Neigh}(v) \cup \{v\},$
- distance degrees $N(v,h) = |\text{Neigh}_h(v)|$
- distance degree sequence $N(v,0), N(v,1), N(v,2), \ldots$
- Hop plot $P(h) = |\{(v,w) : d(v,w) \leq h\}| = \sum_{v \in V} N(v,h)$
  (also called distance distribution)

Applications

What can we do with those $N(v, h)$?

- Compare nodes (their distance degree sequence)
- Rank nodes (which are the "important ones"?)
- Compare graphs (their hop plots)

Exact Algorithm

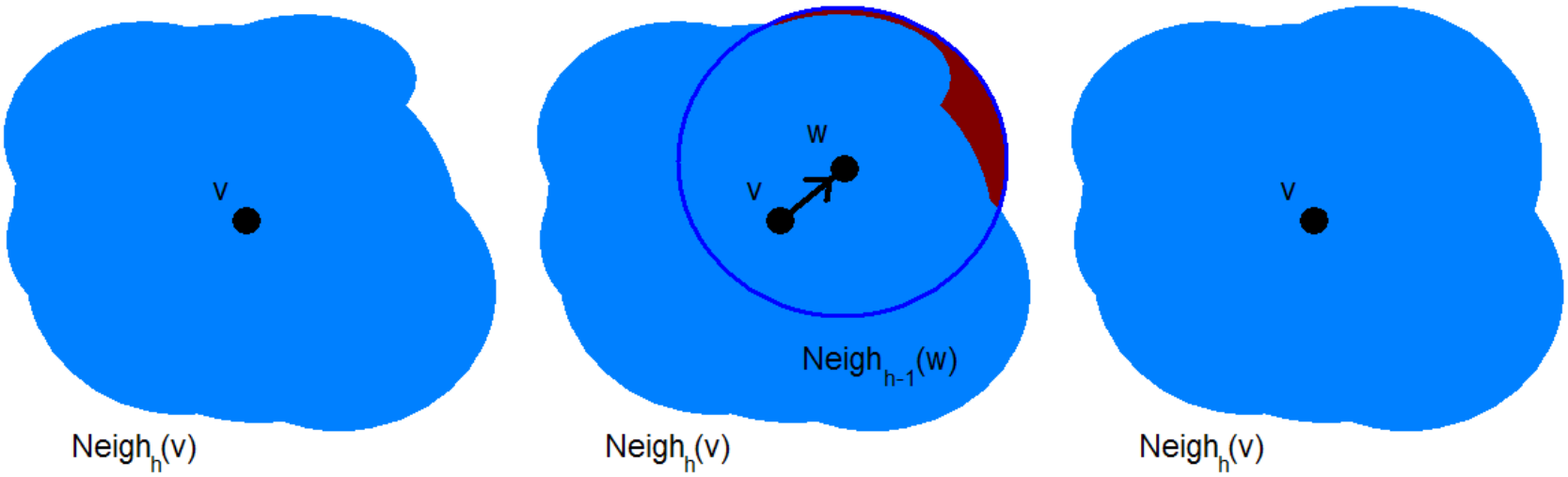
- How can we compute the $N(v, h)$ efficiently for each $v \in V$ and $h = 1, \ldots, \text{diam}(G)$ (even for very large instances)?
- BFS from every vertex?
- No! (random access to edge file)
- Idea: Sequentially scan edge file, grow the set of already reached nodes for each node accordingly
- ANF (Approximate Neighborhood Function) algorithm, Palmer et. al (2002)

Exact Algorithm

Input: Graph $G = (V, E)$
Output: h-neighborhood sizes for all $h \in \mathbb{N}, v \in V$

foreach $v \in V$ **do**
  $\text{Neigh}_0(v) \leftarrow \{v\}$
for $h = 1, \ldots, \text{diam}(G)$ **do**
  foreach $v \in V$ **do**
    $\text{Neigh}_h(v) \leftarrow \text{Neigh}_{h-1}(v)$
  foreach $(v, w) \in E$ **do**
    $\text{Neigh}_h(v) \leftarrow \text{Neigh}_h(v) \cup \text{Neigh}_{h-1}(w)$

Exact Algorithm



Exact Algorithm

- Crucial: Computing the number of distinct elements in

  foreach $(v, w) \in E$ **do**
  $\quad \text{Neigh}_h(v) \leftarrow \text{Neigh}_h(v) \cup \text{Neigh}_{h-1}(w)$

- Maintaining for each node $v \in V$ a bitstring that represents the set of already reached nodes
- Give each node w its own bit in v's bitstring?
- No, needs quadratic space!
- Solution: Approximation to the $N(v, h)$'s by using shorter bit strings

Probabilistic Counting

Probabilistic Counting: Flajolet and Martin (1985)

- Originally designed for data base applications
- Maintain for each node $v \in V$ a bitstring of length $\mathcal{O}(\log n)$
- Throw a node to bit j with probability $\left(\frac{1}{2}\right)^{j+1}$
-

j	0	1	2	3	4	...
probability	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	...

- union of two sets: bitwise OR of the two corresponding bitstrings

Probabilistic Counting, cont'd

How can we estimate the number of elements which are represented by a given bitstring?

- look for the leftmost zero bit (say b)
-

bit	0	1	2	3	4	5	6	...
value	1	1	$\underline{0}$	1	0	0	0	...

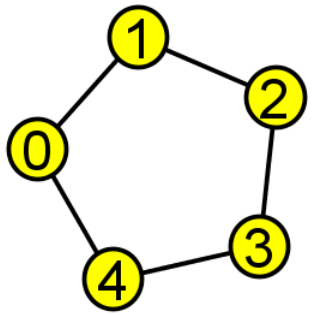
- the number of elements is proportional to 2^b
- proportionality factor $= 0.77351$
- improved accuracy by maintaining k bitstrings and averaging over the resulting b's
- estimation has good provable error bounds!

Basic ANF Algorithm

foreach $v \in V$ **do**
  $M(v, 0) \leftarrow$ concatenation of k bitstrings, each with 1 bit set $(P(i) = \frac{1}{2^{i+1}})$

for $h = 1, \ldots, \text{diam}(G)$ **do**
  foreach $v \in V$ **do**
    $M(v, h) \leftarrow M(v, h-1)$
  foreach $(v, w) \in E$ **do**
    $M(v, h) \leftarrow M(v, h) \vee M(w, h-1)$
  foreach $v \in V$ **do**
    $b \leftarrow$ average position of leftmost zero bits in the k partial bitstrings in $M(v, h)$
    $\widehat{N}(v, h) \leftarrow \frac{2^b}{0.77351}$

Example



Input: a cycle with 5 nodes

$k = 3$

v	$M(v,0)$	$M(v,1)$	$\widehat{N}(v,1)$	$M(v,2)$	$\widehat{N}(v,2)$
0	100 100 001	110 110 101	4.1	110 111 101	5.2
1	010 100 100	110 101 101	3.25	110 111 101	5.2
2	100 001 100	110 101 100	3.25	110 111 101	5.2
3	100 100 100	100 111 100	4.1	110 111 101	5.2
4	100 010 100	100 110 101	3.25	110 111 101	5.2

Example: $\widehat{N}(2,1) = \frac{2^{(2+1+1)/3}}{0.77351} = \frac{2^{4/3}}{0.77351} = 3.25$

Benefits

Why use the ANF algorithm?

- Input (edge file) can stay on disk (sequential access, no random access)
- Scalability, $\mathcal{O}(\text{diam}(G) \cdot m)$ time
- Linear memory usage, $\mathcal{O}(m + n)$
- Can be parallelized
- Good, accurate results (better than sampling etc.)

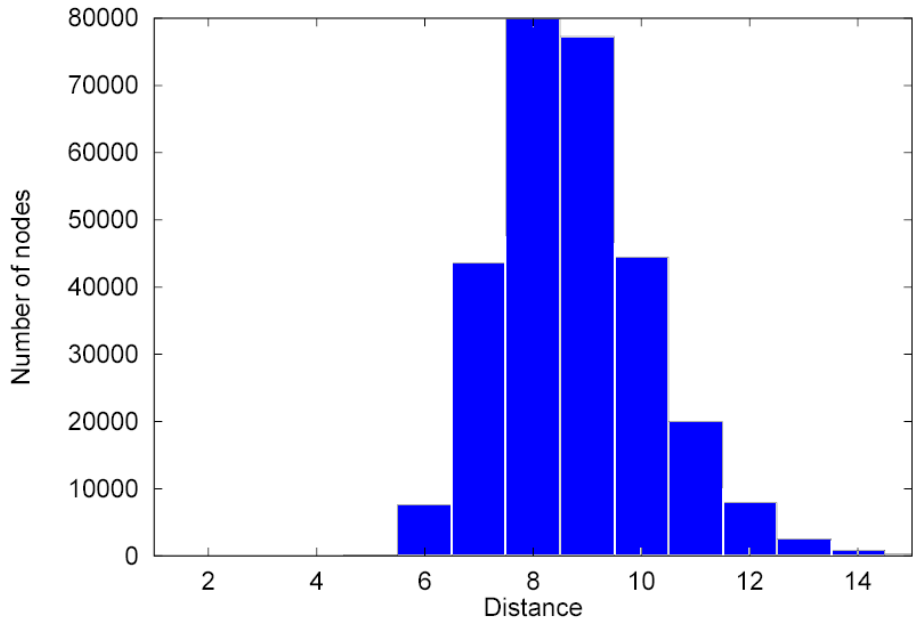
Web Mining

"The Web as a graph"

- Increasing amount of research on graph structure in the WWW
- Objective: get a more global view to the WWW structure
- Typical statistics: average path length, distance distribution, ...

Web Mining

Example: Compute for each node $v \in V$ the minimum distance h such that $N(v, h) \geq \frac{n}{2}$

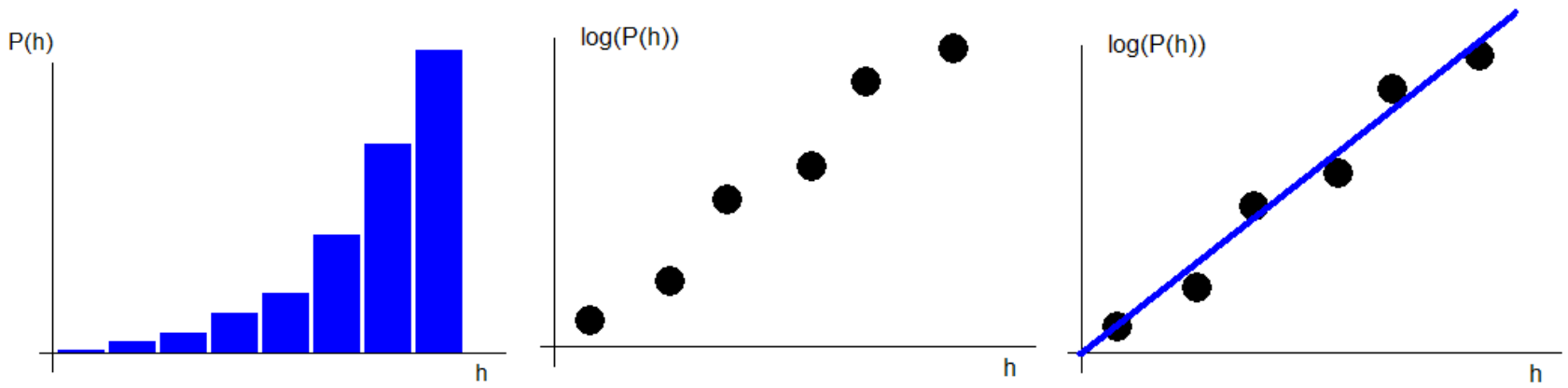


Graph Similarity

Given two graphs, how can we determine their similarity?

- One approach: use the hop plot
  $P(h) = |\{(v, w) : d(v, w) \leq h\}$
- Many real-world graphs seem to have a $P(\cdot)$ following a power law
- $P(h) \sim h^a$, where a is called hop exponent
- Examples: Cycle: $a = 1$, Grid: $a = 2$
- "intrinsic dimensionality" of the graph

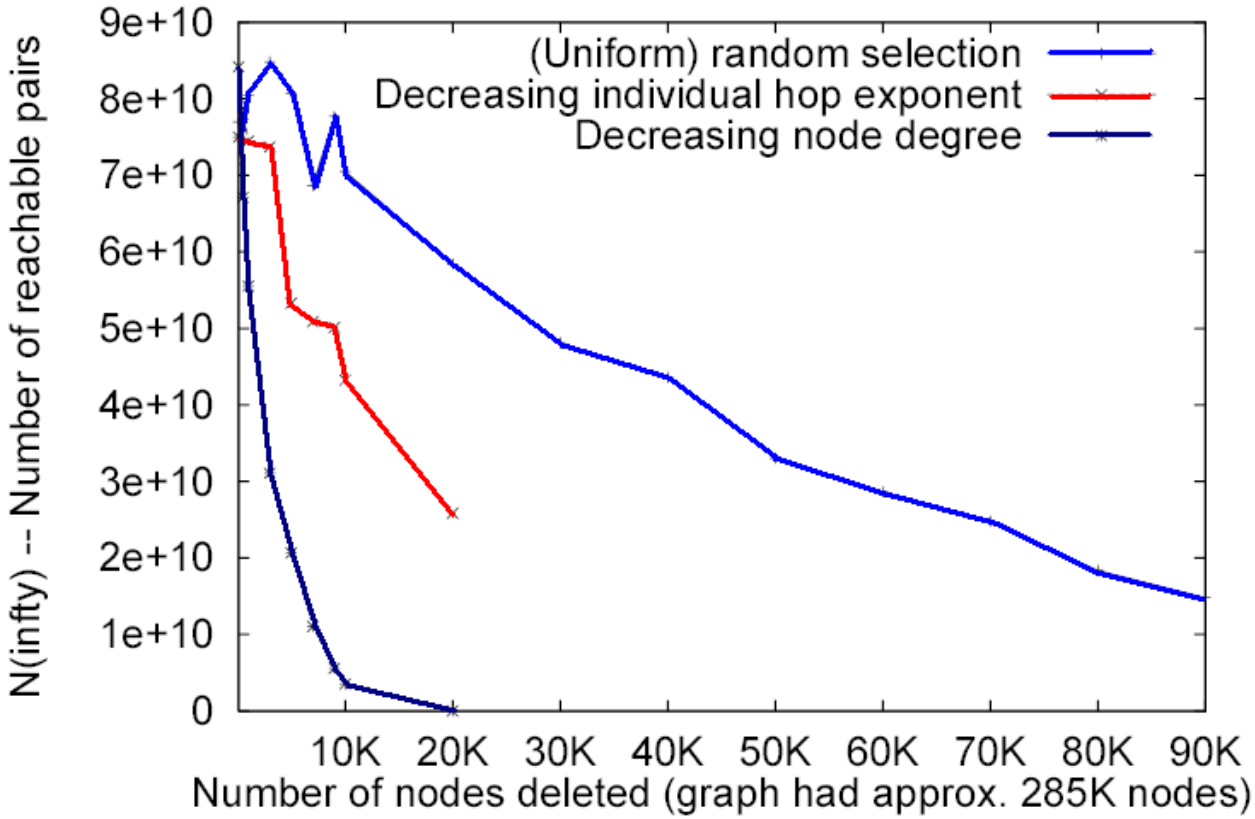
Graph Similarity



Internet Router Data

- Fault-tolerance and connectivity of the internet topology
- Data: Collection of tracert results (285k nodes, 430k edges), pulicly available at www.isi.edu
- Experiments: Successively delete nodes and compute neighborhood information again

Internet Router Data



Summary

Summary

- The h-neighborhoods and the hop plots can be useful to reveal structural properties of the networks
- ANF algorithm yields good approximation to the required information
- Algorithm scales even to very large instances ($>$ 50m nodes)
- Other applications: analysis, clustering, visualization,...