The Web as a graph
measurements, models, and methods

J. Kleinberg, R. Kumar, P. Raghavan,
S. Rajagopalan, A. Tomkins; 1999

Michelle Ackermann <mackerma@student.ethz.ch>
Seminar of Distributed Computing WS 03/04

Overview
- Introduction
- Algorithms
- Measurements
- Model
- Discussion

1. Introduction
- The Web graph is a directed graph of nodes (pages) and directed edges (hyperlinks)
- Several 100 million nodes (grows exponentially in time)
- Today: more than two billion nodes
- Average node has 7 hyperlinks

Reasons to study Web graph
- Improve Web search algorithms
- Topic classification
- Topic enumeration
- Growth of the Web and behavior of users is becoming a serious commercial interest

2. Algorithms
- HITS algorithm searches for high-quality pages on a topic query
- Topic enumeration algorithm enumerates all topics (communities) of the Web graph

Terminology
- Authoritative pages are focused on a particular topic
- Hub pages contain links to relevant pages on the topic

node (pages) and directed edges (hyperlinks)
Hubs Authorities
The HITS algorithm

- Hypertext-induced topic selection
- Reveals the most relevant pages on a search topic
- Sampling step
- Weight-propagation step

Weight-propagation step

- Extract good hubs and authorities from the base set
- Each page $p$ has
  - authority weight $x_p$
  - hub weight $y_p$
- Pages of large hub weights (good hubs) point to pages of large authority weights (good authorities)

Sampling step

- Construct a subgraph expected to be rich in relevant, authoritative pages
- Keyword query to collect root set (~200 pages)
- Expand to base set (1000-3000 pages) by including all pages that link to or are linked by a page in the root set
- Base set contains a large number of authoritative pages

Updating weights

- Increase authority weight if page is pointed to by many good hubs:
  $$x_p = \sum_{i:p} y_i$$
- Increase hub weight if page points to many good authorities:
  $$y_p = \sum_{p:q} x_q$$

More mathematical...

- Adjacency matrix $A$ with entries $(i,j)$:
  - 1 if page $i$ links to page $j$
  - 0 otherwise
- $x = (x_1, x_2, ..., x_n)$
- $y = (y_1, y_2, ..., y_n)$
- new update rules:
  - $x \leftarrow A^T y$
  - $y \leftarrow A x$

...Power iteration

- $x \leftarrow A^T y \leftarrow A^T A x = (A^T A) x$
- $y \leftarrow A x \leftarrow A A^T y = (A A^T) y$
- Multiple iterations $\rightarrow$ Power iteration
  - $k$ iterations $\rightarrow (A^T A)^k x$
  - $x$ converges to principal eigenvector of $A^T A$
Conclusion

- Output list contains
  - pages with the largest hub weights
  - pages with the largest authority weights

- After collecting the root set, the algorithm ignores textual content
  Nevertheless it provides **good search results** for a wide range of queries

---

Topic enumeration

- Enumerates *all* topics (processes entire graph)
- **Bipartite core** $C_{ij}$ contains a complete *bipartite clique* $K_{ij}$

  - Intuition: Every well represented topic will contain a bipartite core $C_{ij}$ for some appropriate $i$ and $j$
  - Enumerate all bipartite cores for some $i$ and $j$

---

Elimination-generation Algorithm

- Number of sequential *passes* over the graph
- Pass consists of elimination and generation
- During each pass, the algorithm writes a modified version of the graph to the disk
- Alternately *sort edges by source and destination* (no random access to edges required)

---

Naive Algorithm

**Problems**

- Size of search space too large
  - $10^8$ nodes $\rightarrow 10^{40}$ possibilities
- Requires random access to edges
  - large fraction of graph must reside in memory

---

Elimination

- Consider example $C_{4,3}$

- Edges of nodes with out-degree smaller 3 can be deleted because the node cannot participate on the left side

- Nodes with in-degree smaller 4 cannot participate on the right side

---

Generation

- **Identify nodes** $u$ that barely qualify for a core
- Either output the core or prove that $u$ doesn’t belong to a core, then drop $u$

- Example: node $u$ with in-degree *exactly* 4 only belongs to a $C_{4,3}$ if the nodes that point to $u$ have a neighborhood intersection of size *at least 3*
Observations

- Experiment: over 90% of the cores are not coincidental and correspond to communities with a definite topic focus
- Challenge: How to organize the discovered communities?
- Other interesting subgraphs: webrings, cliques, directed trees

3. Measurements

- Degree distributions
- Number of bipartite cores
- Connectivity of the graph

We will see that traditional random graph models like $G_{n,p}$ don't explain our observations

Degree distributions

- Measurements show that the in- and out-degrees of the nodes are Zipfian distributed
- Zipfian distribution: probability a node has degree $i$: $P_i \sim 1/i^\alpha$, $\alpha \approx 2$
- $G_{n,p}$ has a binomial degree distribution:

$$P_i = \binom{n}{i} p^i (1 - p)^{n-i} \quad (n = 10^5, p = \frac{7.2}{n})$$

In-degree distribution

Number of bipartite cores

- Experiment:
  - over 100 million pages
  - $i$ ranging from 3 to 6
  - $j$ ranging from 3 to 9
- Result: number of $C_{ij}$ in the Web graph
  - over 100,000 cores
  - $i=3, j=3$: $\approx 40,000$ cores
  - $i=6, j=9$: $\approx 1,000$ cores
Bipartite cores in a random graph

Number of $C_{ij}$ in $G_{n,p}$, $np = 7.2$ (outdegree):

$$\binom{n}{i} \binom{n}{j} p^i (1-p)^{n-i-j} \approx \frac{n^{i+j}}{n^n}$$

which is about 0 for $ij > i + j$

---

Connectivity of the Web

- Bowtie shape
- Strongly connected core (SCC): every page can reach every other by a path (average 20 links)
- IN-pages: can reach the core
- OUT-pages: can be reached by the core
- Scale-free: subgraphs also have the bowtie shape

---

Connectivity of the Web

4. Model

- Reasons for developing a model
- Requirements
  - A class of random graph models

---

Reasons for developing a model

- Model structural properties of the graph
  - degrees
  - distribution of $C_{ij}$
- Predict the behaviour of algorithms on the Web
  - show that an algorithm works well for problems in the model, (but would perform bad on worst-case graphs)
- Make predictions about the shape of the Web graph in the future

---

Requirements

- Model should have an easy and natural description
- Capture aggregate formation of the graph (not detailed individual behaviour)
- Set of topics evolve from the model (no static set required), the Web is dynamic
- Reflect the measurements we have seen
A class of random graph models

- Some page creators link to other sites without regard to existing topics
- Most page creators link to pages within existing topics of interest
  - find resource list for a topic and include many links from the list in the page
  - copying links
- Random copying as a mechanism to create Zipfian degree distributions

Stochastic processes

- Creation processes $C_r$ and $C_e$ \{ discrete time processes
- Deletion processes $D_r$ and $D_e$
- $C_r$ creates a node with probability $a_r(t)$
- $D_r$ removes a node with probability $a_d(t)$ and also deletes all incident edges
- $D_e$ deletes an edge with probability $\delta(t)$
- Choose probabilities to reflect growth rates of the Web, half-life of pages, etc.

Edge creation process

- Determine a node $v$ and a number $k$
- With probability $\beta$ add edges pointing to $k$ uniformly chosen nodes
- With probability $1-\beta$ copy $k$ edges from a randomly chosen node $u$
- If the outdegree of $u$ is more than $k$, choose a random subset of size $k$
- If the outdegree of $u$ is less than $k$, copy the edges and choose another node $u'$

A simple model

- New node created at every time step
- No deletions
- Choose $u$ uniformly at random
- $\beta$: new edge points to $u$
- $1-\beta$: copy the out-link from $u$

Simulation

- Probability a node has indegree $i$
  converges to $1/i^\alpha$, $\alpha=1/(1-\beta)$
- Number of cores significantly larger than in a traditional random graph

Challenges

- Study relationship between copying and Zipfian distributions (applications outside the Web: term frequencies, genome, etc.)
- Study properties and evolution of the random graphs generated by the model
- Need efficient algorithms to analyze such graphs because copying generates myriads of dependencies