On Compressing Social Networks

Ravi Kumar
Yahoo! Research, Sunnyvale, CA
Joint work with

- Flavio Chierichetti, University of Rome
- Silvio Lattanzi, University of Rome
- Michael Mitzenmacher, Harvard
- Alessandro Panconesi, University of Rome
- Prabhakar Raghavan, Yahoo! Research
Behavioural graphs

- Web graphs
- Host graphs
- Social networks
- Collaboration networks
- Sensor networks
- Biological networks
- ...

Research trends
- Empirical analysis: examining properties of real-world graphs
- Modeling: finding good models for behavioural graphs

There has been a tendency to lump together behavioural graphs arising from a variety of contexts
Properties of behavioural graphs

- Degree distributions
  - Heavy tail
- Clustering
  - High clustering coefficient
- Communities and dense subgraphs
  - Abundance; locally dense, globally sparse; spectrum
- Connectivity
  - Exhibit a “bow-tie” structure; low diameter; small-world properties
A remarkable empirical fact

- Snapshots of the web graph can be compressed using less than 3 bits per edge
  Boldi, Vigna WWW 2004

- Improved to ~2 bits using another data mining inspired compression technique
  Buehrer, Chellapilla WSDM 2008

- More recent improvements
  Boldi, Santinin, Vigna WAW 2009

Key insights
1. Many web pages have similar set of neighbors
2. Edges tend to be “local”
Are social networks compressible?

- Review of BV compression
- A different compression mechanism that works better for social networks
- A heuristic
- its performance
- and a formalization
Why study this question?

- Efficient storage
  - Serve adjacency queries efficiently in-memory
  - Archival purposes — multiple snapshots

- Obtain insights
  - Compression has to utilize special structure of the network
  - Study the randomness in such networks
Adjacency table representation

- Each row corresponds to a node \( u \) in the graph.
- Entries in a row are sorted integers, representing the neighborhood of \( u \), i.e., edges \((u, v)\).

1: 1, 2, 4, 8, 16, 32, 64
2: 1, 4, 9, 16, 25, 36, 49, 64
3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
4: 1, 4, 8, 16, 25, 36, 49, 64

- Can answer adjacency queries fast.
- Expensive (better than storing a list of edges).
Boldi-Vigna (BV): Main ideas

- **Similar neighborhoods**: The neighborhood of a web page can be expressed in terms of other web pages with similar neighborhoods
  - Rows in adjacency table have similar entries
  - Possible to choose to **prototype** row

- **Locality**: Most edges are intra-host and hence local
  - Small integers can represent edge destination wrt source

- **Gap encoding**: Instead of storing destination of each edge, store the difference from the previous entry in the same row

```plaintext
1: 1, 2, 4, 8, 16, 32, 64
2: 1, 4, 9, 16, 25, 36, 49, 64
3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
4: 1, 4, 8, 16, 25, 36, 49, 64
```
Finding similar neighborhoods

Canonical ordering: Sort URLs lexicographically, treating them as strings

...  
17: www.stanford.edu/alchemy  
18: www.stanford.edu/biology  
19: www.stanford.edu/biology/plant  
20: www.stanford.edu/biology/plant/copyright  
21: www.stanford.edu/biology/plant/people  
22: www.stanford.edu/chemistry  
...

This gives an identifier for each URL

Source and destination of edges are likely to get nearby IDs
  - Templated webpages
  - Many edges are intra-host or intra-site
**Gap encodings**

- Given a sorted list of integers $x, y, z, \ldots$, represent them by $x, y-x, z-y, \ldots$

- Compress each integer using a code
  - $\gamma$ code: $x$ is represented by concatenation of unary representation of $\lceil \lg x \rceil$ (length of $x$ in bits) followed by binary representation of $x - 2^{\lceil \lg x \rceil}$
    
    Number of bits = $1 + 2^{\lceil \lg x \rceil}$

- $\delta$ code: ...

- Information theoretic bound: $1 + \lceil \lg x \rceil$ bits

- $\zeta$ code: Works well for integers from a power law
  
  Boldi Vigna DCC 2004
BV compression

Each node has a unique ID from the canonical ordering
Let \( w = \) copying window parameter
To encode a node \( v \)
- Check if out-neighbors of \( v \) are similar to any of \( w-1 \) previous nodes in the ordering
- If yes, let \( u \) be the prototype: use \( \lg w \) bits to encode the gap from \( v \) to \( u + \) difference between out-neighbors of \( u \) and \( v \)
- If no, write \( \lg w \) zeros and encode out-neighbors of \( v \) explicitly
Use gap encoding on top of this

\[ \leq w-1 \]
Main advantages of BV

- Depends only on locality in a canonical ordering
  - Lexicographic ordering works well for web graph

- Adjacency queries can be answered very efficiently
  - To fetch out-neighbors, trace back the chain of prototypes until a list whose encoding beings with $\text{lg}$ w zeros is obtained (no-prototype case)
  - This chain is typically short in practice (since similarity is mostly intra-host)
  - Can also explicitly limit the length of the chain during encoding

- Easy to implement and a one-pass algorithm
Social networks are highly reciprocal, despite being directed.

- If A is a friend of B, then it is likely B is also A’s friend.

- \((u, v)\) is **reciprocal** if \((v, u)\) also exists.

\[\text{reciprocal}(u) = \text{set of } v\text{'s such that } (u, v)\text{ is reciprocal}\]

- **How to exploit reciprocity in compression?**
  - Can avoid storing reciprocal edges twice.
  - Just the reciprocity “bit” is sufficient.
Backlinks compression (contd)

Given a canonical ordering of nodes and copying window $w$

To encode a node $v$

- encode out-degree of $v$ minus 1 (if self loop) minus $\#\text{reciprocal}(v) + \text{“self-loop” bit}$

- Try to choose a prototype $u$ as in BV within a window $w$

- If yes, encode the difference between out-neighbors of $u$ and non-reciprocal out-neighbors of $v$
  - Encode the gap between $u$ and $v$
  - Specify which out-neighbors of $u$ are present in $v$
  - For the rest of out-neighbors of $v$, encode them as gaps

- Encode the reciprocal out-neighbors of $v$
  - For each out-neighbor $v'$ of $v$ and $v' > v$, store if $v' \in \text{reciprocal}(v)$ or not; discard the edge $(v', v)$
Canonical orderings

- BV and BL compressions depend just on obtaining a canonical ordering of nodes
  - This canonical ordering should exploit neighborhood similarity and edge locality

- **Question:** how to obtain a good canonical ordering?
  - Unlike the web page case, it is unclear if social networks have a natural canonical ordering

- **Caveat:** BV/BL is only one genre of compression scheme
  - Lack of good canonical ordering does not mean graph is incompressible
Some canonical orderings in behavioral graphs

- Random order
- Natural order
  - Time of joining in a social network
  - Lexicographic order of URLs
  - Crawl order
- Graph traversal orders
  - BFS and DFS
- Geographic location: order by zip codes
  - Produces a bucket order
- Ties can be broken using more than one order
Performance of simple orderings

<table>
<thead>
<tr>
<th>Graph</th>
<th>#nodes</th>
<th>#edges</th>
<th>%reciprocal edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr</td>
<td>25.1M</td>
<td>69.7M</td>
<td>64.4</td>
</tr>
<tr>
<td>UK host graph</td>
<td>0.58M</td>
<td>12.8M</td>
<td>18.6</td>
</tr>
<tr>
<td>IndoChina</td>
<td>7.4M</td>
<td>194.1M</td>
<td>20.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BV</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Natural</td>
<td>Random</td>
<td>DFS</td>
<td></td>
</tr>
<tr>
<td>Flickr</td>
<td>21.8</td>
<td>23.9</td>
<td>22.9</td>
<td></td>
</tr>
<tr>
<td>UK host</td>
<td>10.8</td>
<td>15.5</td>
<td>14.6</td>
<td></td>
</tr>
<tr>
<td>IndoChina</td>
<td>2.02</td>
<td>21.44</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BL</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Natural</td>
<td>Random</td>
<td>DFS</td>
<td></td>
</tr>
<tr>
<td>Flickr</td>
<td>16.4</td>
<td>17.8</td>
<td>17.2</td>
<td></td>
</tr>
<tr>
<td>UK host</td>
<td>10.5</td>
<td>14.5</td>
<td>13.8</td>
<td></td>
</tr>
<tr>
<td>IndoChina</td>
<td>2.35</td>
<td>17.6</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Shingle ordering heuristic

- Obtain a canonical ordering by bringing nodes with similar neighborhoods close together.
- Fingerprint neighborhood of each node and order the nodes according to the fingerprint.
  - If fingerprint can capture neighborhood similarity and edge locality, then it will produce good compression via BV/BL, provided the graph has amenable BV/BL.
- Use Jaccard coefficient to measure similarity between nodes.
  - \[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} \]
A fingerprint for Jaccard

Fingerprint to measure set overlap

\[ M_\pi(A) = \min_{a \in A} \{ \pi(a) \} \]

\[ \Pr_\pi [M_\pi(A) = M_\pi(B)] = \frac{|A \cap B|}{|A \cup B|} \]

Min-wise independent permutations suffice
Broder, Charikar, Frieze, Mitzenmacher STOC 1998
Hash functions work well in practice
Shingle ordering heuristic (contd)

- Fingerprint of a node $u = M_\pi(\text{out-neighbors of } u)$
- Order the nodes by their fingerprint
  - Two nodes with lot of overlapping neighbors are likely to have same shingle

- **Double shingle order**: break ties within shingle order using a second shingle
## Performance of shingle ordering

<table>
<thead>
<tr>
<th>Graph</th>
<th>Natural</th>
<th>Shingle</th>
<th>Double shingle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr</td>
<td>21.8</td>
<td>13.5</td>
<td>13.5</td>
</tr>
<tr>
<td>UK host</td>
<td>10.8</td>
<td>8.2</td>
<td>8.1</td>
</tr>
<tr>
<td>IndoChina</td>
<td>2.02</td>
<td>2.7</td>
<td>2.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graph</th>
<th>Natural</th>
<th>Shingle</th>
<th>Double shingle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr</td>
<td>16.4</td>
<td>10.9</td>
<td>10.9</td>
</tr>
<tr>
<td>UK host</td>
<td>10.5</td>
<td>8.2</td>
<td>8.1</td>
</tr>
<tr>
<td>IndoChina</td>
<td>2.35</td>
<td>2.7</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Geography does not seem to help for Flickr graph.
Flickr: Compressibility over time
Theorem. Using shingle ordering, a constant fraction of edges will be “copied” in graphs generated by preferential attachment/copying models.

- Preferential attachment model: Rich get richer — a new node links to an existing node with probability proportional to its degree.
- Shows that shingle ordering helps BV/BL-style compressions in stylized graph models.
Gap distribution

Shingle ordering produces smaller gaps
Who is the culprit

Low degree nodes are responsible for incompressibility
Computation-friendly orderings

In BV/BL, canonical order is all that matters

Problem. Given a graph, find the canonical ordering that will produce the best compression in BV/BL

- The ordering should capture locality and similarity
- The ordering must help BV/BL-style compressions

- We propose two formulations of this problem
MLogA formulation

MLogA. Find an ordering \( \pi \) of nodes such that

\[
\sum_{(u, v) \in E} \lg |\pi(u) - \pi(v)|
\]

is minimized

- Minimize sum of encoding gaps of edges
- Without \( \lg \), this is min linear arrangement (MLinA)
- MLinA is well-studied (\( (\sqrt{\log n}) \log \log n \)) approximable, ...
- MLinA and MLogA are very different problems

Theorem. MLogA is NP-hard

Proof using the inapproximability of MaxCut
MLogGapA formulation

MLogGapA. For an ordering \( \pi \), let \( f_\pi(u) = \text{cost of compressing the out-neighbors of } u \text{ under } \pi \)

If \( u_1, \ldots, u_k \) are out-neighbors ordered wrt \( \pi \), \( u_0 = u \)

\[
f_\pi(u) = \sum_{i=1}^{k} \lg |\pi(u_i) - \pi(u_{i-1})|\]

Find an ordering \( \pi \) of nodes to minimize

\[
\sum_u f_\pi(u)
\]

- Minimize encoding gaps of neighbors of a node
- MLogGapA and MLogA are very different problems

Theorem. MLinGapA is NP-hard

Conjecture. MLogGapA is NP-hard
Summary

- Social networks appear to be not very compressible
- Host graphs are equally challenging
- These two graphs are very unlike the web graph, which is highly compressible
Future directions

- Can we compress social networks better?
  - Boldi, Santini, Vigna 2009

- Is there a lower bound on incompressibility?
  - Our analysis applies only to BV-style compressions

- Algorithmic questions
  - Hardness of MLogGapA
  - Good approximation algorithms

- Modeling
  - Compressibility of existing graph models
  - More nuanced models for the compressible web
  - Chierichetti, Kumar, Lattanzi, Mitzenmacher, Panconesi, Raghavan FOCS 2009
Thank you!

ravikumar@yahoo-inc.com