Compressibility of Behavioral Graphs

Ravi Kumar
Google, Mountain View, CA
ravi.k53@gmail.com
Behavioral 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 behavioral graphs

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

- Heavy-tail degree distributions, eg, power law $p(x) \propto x^{-\alpha}$
Other structural properties

- **Clustering**
  - High clustering coefficient

- **Communities and dense subgraphs**
  - Abundance; locally dense, globally sparse

- **Connectivity**
  - Exhibit a “bow-tie” structure; low diameter; small-world properties
A remarkable empirical fact

- Snapshots of the web graph can be losslessly 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

- Subsequent improvements
  Boldi, Santini, Vigna WAW 2009

<table>
<thead>
<tr>
<th>18.5 Mpages, 300 Mlinks from .uk</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>$\infty$</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>$\infty$</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>118 Mpages, 1 Glinks from WebBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>$\infty$</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Traspose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>$\infty$</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Why study compressibility?

- **Efficient storage**
  - Serve adjacency queries in-memory – enables efficient algorithms
  - Archival purposes – store multiple snapshots efficiently

- **Obtain new insights**
  - Compression captures global network structure
  - Study the randomness in behavioral graphs
  - Validate existing graph models

- **Algorithmic considerations**
  - Possibility of working directly on compressed representations [Karande, Chellapilla, Andersen WSDM 2009]
Adjacency list representation

- Each row corresponds to a node $u$ in the graph
- Entries in a row are sorted integers, representing the neighborhood of $u$, ie, 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 to store
  
  Though, better than storing a list of edges
Neighborhood similarity

- **Similar neighborhoods**: 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 a *leader* 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
  - **Distribution of gap values**: Optimal codes

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
The Boldi-Vigna scheme

Boldi-Vigna get down to an average of \(~3\) bits/URL-URL edge, for an 118M node web graph

- **How does it work?**
- **Why does it work?**
Main ideas of Boldi-Vigna

Canonical ordering: Sort URLs alphabetically, treating them as strings Randall et al 2002

...  
17: www.berkeley.edu/alchemy  
18: www.berkeley.edu/biology  
19: www.berkeley.edu/biology/plant  
20: www.berkeley.edu/biology/plant/copyright  
21: www.berkeley.edu/biology/plant/people  
22: www.berkeley.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
Due to templates, the adjacency list of a node is similar to one of the 7 preceding URLs in the alphabetic ordering.

Express adjacency list in terms of one of these.

Eg, consider these adjacency lists:

- **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

Encode as (-2), remove 9, add 8.
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 $\lfloor \log_2 x \rfloor$ (length of $x$ in bits) followed by binary representation of $x - 2^{\lfloor \log_2 x \rfloor}$
    
    Number of bits = $1 + 2^{\lfloor \log_2 x \rfloor}$
  - $\delta$ code: ...
  - Information theoretic bound: $1 + \lfloor \log_2 x \rfloor$ bits
  - $\zeta$ code: Works well for integers from a power law [Boldi, Vigna DCC 2004]
BV compression algorithm

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 leader: 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
Main advantages of BV

- Depends only on locality in a canonical ordering
  - Alphabetic ordering works well for web graph
- Adjacency queries can be answered very efficiently
  - To fetch out-neighbors, trace back the chain of leaders until a list whose encoding beings with \( \lg w \) zeros is obtained (no-leader 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
Practice vs Theory

Why does Boldi-Vigna compression work?
- **Similarity**: Many nodes have similar neighborhoods
- **Locality**: Most edges are local

Graph models and compression
- Are graphs generated by existing models compressible?
- Can we formulate a model with locality?

Social networks and compression
- Are social networks as compressible as the Web?
- Popular papers are cited more
- Popular people are befriended more

Each step has one new incoming node along with an edge

Probability this new node links to a pre-existing node is proportional to how popular is the latter, ie, its degree

\[ \Pr[\text{new node links to node } i] = \frac{d_i}{\sum d_j} \]

Theorem. Degree distribution is a power law with exponent 3

Intuitive proof. \( \frac{\partial d_i}{\partial t} = \frac{d_i}{2t} \)

If node i was added at time \( t_i \), then \( d_i(t) = \left(\frac{t}{t_i}\right)^{0.5} \)

\[ \Pr[d_i(t) > k] = \Pr[t_i < t/k^2] = \frac{1}{k^2} \]
Other “non-local” models

- **Copying model** [Kumar et al. FOCS 2000]
  - **Observation**: People copy their friend’s webpage when creating a new one or copy their friend’s contacts when joining a social network.
  - When a new node arrives, it copies edges from a pre-existing node with probability $1 - \alpha$.
  - The degree distribution is a power-law with exponent $\frac{(2 - \alpha)/(1 - \alpha)}{2}$.
  - Can explain communities: The number of dense bipartite cliques in this model is large.

- **Forest-fire model** [Leskovec, Kleinberg, Faloutsos KDD 2005]
  - An iterated version of the copying model.
  - In addition to the above, leads to densification and shrinking diameters.
**Incompressibility** Chierichetti et al FOCS 2009

**Theorem.** The following generative models all require \( \Omega(\log n) \) bits per edge on average, even if the node labels are removed:

- the **preferential attachment** model
- the **copying** model
- the evolutionary **ACL** model Aiello, Chung, Lu FOCS 2001
- **Kronecker** multiplication model Leskovec et al PKDD 2005
- Model for navigability in social networks Kleinberg Nature 2000

- We remove labels since BV compresses unlabeled Web graphs to \( O(1) \) bits per edge

- **Min-entropy argument:** Find a subset of graphs
  - **not too large:** to avoid graphs that are “easy”
  - **not too small:** should still contain interesting graphs about which we can show incompressibility
A new graph model

- Begin with a seed graph of nodes with out-degree $k$, arranged in a cycle
- Additional nodes arrive in sequence
- An arriving node is inserted before a random node in the cycle (leader)
  - It links to $k-1$ out-neighbors of its leader
  - It links to the leader
An example, $k=2$
Locality in the new model

- If a web designer wants to add a new web page to her web site
  - likely to take some existing web page on her website
  - modify it as needed (perturbing the set of its outlinks) to obtain the new page
  - adding a reference to the old web page
  - and publish the new web page on her website

- Since web pages are sorted by URL in our ordering, the old and the new page will be close!
Basic properties of the model

- Rich get richer: in the model, in-degrees converge to a power law with exponent $-2-1/(k-1)$
- High clustering coefficient
- Polynomially many bipartite cliques
- Logarithmic undirected diameter
- Compressible to $O(1)$ bits per edge
- In fact, BV algorithm achieves $O(1)$ bits per edge
Theorem. The number of bits required by BV algorithm is $\sum_{l=1}^{\infty} Y_l \log l$, where $Y_l$ is the number of edges of length $l$.

Theorem. In the model, edge lengths converge to a power law with exponent $-1/k$.

Corollary. The new model produces graphs compressible to $O(1)$ bits per edge.
Long gets longer

- Recall the process: pick a leader node uniform at random and place new node to its immediate left
- The probability to become longer is proportional to the number of nodes “below” the edge, i.e., its length
- Making this precise requires pinning down subtle combinatorial properties of the model
Are social networks compressible?

- How does BV perform on social networks?
- Can we take use special properties, eg, 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
- How to exploit reciprocity in compression?
  - Can avoid storing reciprocal edges twice
  - Just the reciprocity “bit” is sufficient
  - Modify BV to get a new scheme
Canonical orderings

- BV compressions depend on a canonical ordering of nodes
  - This canonical ordering should exploit neighborhood similarity and edge locality
- How do we get a good canonical ordering?
  - Unlike the web page case, it is unclear if social networks have a natural canonical ordering
- Caveat: BV is only one genre of compression scheme
  - Lack of good canonical ordering does not mean graph is incompressible
Some natural canonical orderings

- Random order
- Natural order
  - Time of joining in a social network
  - Lexicographic order of URLs
  - Crawl order
- Graph traversal orders
  - BFS and DFS
- Use attributes of the nodes
  - Eg, Geographic location: order by zip codes
  - May produce 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>Graph</th>
<th>Natural</th>
<th>Random</th>
<th>DFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr</td>
<td>21.8</td>
<td>23.9</td>
<td>22.9</td>
</tr>
<tr>
<td>UK host</td>
<td>10.8</td>
<td>15.5</td>
<td>14.6</td>
</tr>
<tr>
<td>IndoChina</td>
<td>2.02</td>
<td>21.44</td>
<td>-</td>
</tr>
</tbody>
</table>
Detour: Shingles

- **Jaccard coefficient**: Measures similarity between sets $A$ and $B$
  $$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
- $1 - J(A, B)$ is a metric
Can we construct a hash function $h$ such that

$$\Pr[h(A) = h(B)] = \frac{|A \cap B|}{|A \cup B|} = J(A, B)$$

- Given a universe $U$, pick a permutation $\pi$ on $U$ uniformly at random.
- Hash each subset $S \subseteq U$ to the minimum value it contains according to $\pi$. 
Shingle ordering heuristic

- Chierichetti et al KDD 2009
- Obtain a canonical ordering by bringing nodes with similar neighborhoods close together
- **Fingerprint neighborhood of each node**
  - Order the nodes according to the fingerprint
  - If fingerprint can capture neighborhood similarity and edge locality, then it can produce good compression via BV
- **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>

**Geography does not seem to help for Flickr graph**
Flickr: Compressibility over time
A property of shingle ordering

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-style compressions in stylized graph models
Who is the culprit

Low degree nodes are responsible for incompressibility
Compression-friendly orderings
Chierichetti et al KDD 2009

In BV, canonical order is all that matters

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

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

- We propose a formulation of this problem

- Recent developments
  - Gray-code ordering Boldi, Santini, Vigna IM 2010
  - Multi-scale ordering Safro, Temkin JDA 2010
  - Layered Label Propagation Boldi, Rosa, Santini, Vigna WWW 2011
**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

**Theorem.** MLinGapA is NP-hard

**Conjecture.** MLogGapA is NP-hard
Summary

- Social networks appear to be not very compressible, but the Web graph is
  - Both exhibit “local” power laws
  - Host graphs are equally challenging

- BV compression
  - Optimal orderings
  - Combinatorial formulations and heuristics

- Generative models
  - Lower bounds for prior models
  - New compressible model
Future directions

- Can we compress social networks better?
- Is there a lower bound on incompressibility?
  - Our analysis applies only to BV-style compressions
- Algorithmic questions
  - Hardness of MLogGapA
  - Good approximation algorithms for good orderings
  - Algorithms that work on compressed graphs
- Modeling questions
  - More nuanced, tractable models for compressibility
Thank you!

ravi.k53@gmail.com